Sistemas de localización avanzados en entornos interiores basados en tecnología UWB

Autor: Miguel Eguizábal Alonso

Director: Juan Chóliz Muniesa

Ponente: Ángela Hernández Solana

Curso 2009/2010

1 de Septiembre de 2010

Ingeniería Superior de Telecomunicación

Departamento de Ingeniería Electrónica y Comunicaciones

Centro Politécnico Superior

Universidad de Zaragoza

Sistemas de localización avanzados en entornos interiores basados en tecnología UWB

Resumen

El presente proyecto fin de carrera, se ha realizado en el marco del proyecto integrado EUWB dentro del VII Programa Marco de la Unión Europea, en el que participa la Universidad de Zaragoza a través del Grupo de Tecnologías de las Comunicaciones (GTC) del Instituto de Investigación en Ingeniería de Aragón (I3A).

La localización de los terminales móviles se ha convertido en un requisito importante para las redes inalámbricas comerciales. Mientras que en entornos exteriores la tecnología GPS está ampliamente extendida, en entornos interiores no existe ninguna tecnología predominante. La tecnología Ultra WideBand (UWB) se caracteriza por la utilización de pulsos de muy corta duración, lo que le permite localizar terminales móviles con un error del orden de las decenas de centímetros, midiendo el retardo de un pulso desde que es emitido por el transmisor hasta que llega al receptor. Además, otras ventajas como su bajo consumo, su inmunidad frente al efecto multi-camino, o la posibilidad de simultanear transmisión de información y localización, hacen de UWB una tecnología prometedora para aplicaciones de sistemas de localización en interiores.

Este PFC se centra en la implementación y evaluación de diferentes algoritmos avanzados de localización, de diversas complejidades y en particular de los propuestos en el proyecto EUWB. Para ello se ha empleado un simulador de localización en entornos interiores en C++ ya existente, realizando modificaciones para adaptarlo a los requisitos de este proyecto. Algunas modificaciones que se han realizado para tratar de mejorar el comportamiento de los algoritmos de localización son el prefiltrado y la ponderación de las distancias estimadas. Para la evaluación de los algoritmos se han llevado a cabo simulaciones en diferentes escenarios y situaciones.

Además, se plantea también la mejora de algunos de los algoritmos considerados a través de la utilización de información geográfica y estadística, de cara a optimizar el compromiso entre precisión y recursos utilizados. En particular, estas mejoras se centran en dos aspectos, que son la detección de situaciones NLOS y la utilización de información acerca de las rutas de los targets.

Índice

1. Descripción del proyecto

1.1 Introducción

El conocimiento acerca de la localización de los terminales móviles se ha convertido en un requisito importante para las redes inalámbricas comerciales, de servicio público y militares. Mientras que en entornos exteriores, la tecnología GPS está ampliamente extendida con aplicaciones como los sistemas de navegación para automóviles, la gestión de flotas o la localización de llamadas de emergencia, las potenciales aplicaciones de la localización en interiores no se están explotando debido a la incapacidad de GPS para operar en entornos interiores. Esta situación ha dado lugar a la aparición de numerosas tecnologías que tratan de cubrir esas aplicaciones de localización en interiores.

La precisión y el alcance son aspectos importantes en los sistemas de localización y dependen en gran medida de las señales y por tanto, de la tecnología utilizada. Algunas de las tecnologías utilizadas para proporcionar localización en interiores son UWB, ZigBee, WLAN, Bluetooth o redes de acceso celular como GSM.

En general, la localización se compone de dos fases, la estimación de las distancias, también denominada ranging, y el cálculo de la posición. La estimación de las distancias se basa en la medida de distintos parámetros como pueden ser el ángulo de llegada (AOA, Angle of Arrival), el nivel de señal recibida (RSSI, Received Signal Strength Indication) o el tiempo de llegada (TOA, Time of Arrival) de señales de referencia transmitidas entre el elemento a localizar y los nodos de referencia. Por otro lado, para el cálculo de la posición existen multitud de algoritmos, desde los más sencillos basados en cálculos geométricos hasta algoritmos de seguimiento que tienen en cuenta la posición anterior del usuario y su trayectoria.

1.1.1 La tecnología UWB

Ultra wideband (UWB) es una tecnología que nació durante la década de 1960, y cuyo nombre fue acuñado por el Departamento de Defensa de los Estados Unidos en 1989. Se desarrolló para radar, localización y aplicaciones de comunicaciones. Se definen como tecnología UWB aquellos sistemas en los que el ancho de banda de la transmisión ocupa más de un 20% de la frecuencia central o más de 500 MHz.

En general los sistemas UWB usan señales basadas en trenes de pulsos de corta duración. El intervalo entre los pulsos individuales puede ser uniforme o variable y existen diferentes métodos para modular el tren de pulsos con datos para comunicaciones. Otro punto importante en los sistemas UWB es que los pulsos individuales son muy cortos de duración, normalmente mucho más cortos que el intervalo correspondiente a la duración de un bit, por lo que la señal resultante tiene un ancho de banda grande.

Debido a sus características, esta tecnología permite localizar los terminales móviles con un error del orden de las decenas de centímetros. Debido a que UWB está basada en pulsos ultracortos, el receptor puede determinar el tiempo de llegada con precisión de picosegundos y, por tanto, estimar la posición con precisión de centímetros. La distancia al

móvil se calcula midiendo el retardo de un pulso desde que es emitido por el transmisor hasta que llega al receptor. Posteriormente, utilizando algoritmos de posicionamiento se determina con gran exactitud la posición del terminal.

Algunas ventajas que presenta UWB son:

- Estimación de posición con precisión de centímetros.
- Grandes velocidades de transmisión con una baja potencia transmitida ya que emplea un gran ancho de banda.
- Señal inmune al efecto multi-camino ya que, debido a su corta duración, es posible distinguir entre la señal directa y las señales reflejadas en los obstáculos.

Centrándose en las aplicaciones de la tecnología UWB, cabe distinguir entre dos tipos fundamentales de sistemas UWB:

- HDR/VHDR (High/Very High Data Rate): Presentan tasas de transmisión elevadas, hasta centenares de Mbps, y alcances cortos, hasta 5 metros. No se basan en sistemas pulsados, sino en modulaciones como Multiband-OFDM y DS-SS (Direct Sequence Spread Spectrum). Se estudiaron en el grupo IEEE 802.15.3a, y la propuesta basada en MB-OFDM ha sido estandarizada por la WiMedia Alliance.
- LDR/LDR-LT (Low Data Rate with Location Tracking): Presentan tasas de transmisión inferiores, hasta la decena de Mbps, pero con mayores alcances, del orden de decenas de metros. Además permiten una gran resolución en la estimación de la distancia, por lo que son apropiadas para la localización y seguimiento. El uso de UWB en redes inalámbricas de área personal (WPANs) de baja tasa se contempla en el estándar 802.15.4a.

A modo de documentación se utilizaron las referencias [1], [2] y [3].

1.2 Marco del proyecto

El presente proyecto fin de carrera, se ha realizado en el marco del proyecto integrado EUWB dentro del VII Programa Marco de la Unión Europea, en el que participa la Universidad de Zaragoza a través del Grupo de Tecnologías de las Comunicaciones (GTC) del Instituto de Investigación en Ingeniería de Aragón (I3A).

Dentro del proyecto EUWB, la participación del grupo investigador de la Universidad de Zaragoza se focaliza en dos grupos de trabajo: el grupo de trabajo 4, centrado en la investigación de sistemas de localización y seguimiento avanzados, y el grupo de trabajo 6, centrado en la integración de UWB y otras tecnologías en redes heterogéneas.

Uno de los posibles escenarios dentro del grupo de trabajo de redes heterogéneas es la integración de la tecnología LDR-LT UWB en dispositivos de usuario en redes de acceso inalámbricas tales como UMTS o WiMAX. El objetivo de integrar UWB en estos dispositivos es el de proporcionar información de localización en entornos interiores, complementando la

información de posicionamiento en exteriores proporcionada por GPS. Los principales escenarios de aplicación identificados son áreas de interiores relativamente grandes y con una alta densidad de usuarios tales como centros comerciales, palacios de congresos, aeropuertos, etc. La disponibilidad de información de posicionamiento es fundamental para el desarrollo de servicios basados en localización, además de poder ser aprovechada por la propia red de acceso celular para mejorar la gestión de recursos radio.

Dentro del grupo de trabajo 4, una de las tareas desarrolladas por el grupo investigador de la Universidad de Zaragoza es el desarrollo de una herramienta de simulación que permita analizar las prestaciones reales de un sistema de localización UWB en un entorno interior de área relativamente grande, evaluando las posibles alternativas de diseño (arquitecturas, algoritmos, etc.). Para ello se utilizan las especificaciones reales tanto a nivel físico como de acceso al medio de los dispositivos LDR-LT UWB que se han desarrollado dentro del propio proyecto. El presente PFC se enmarca dentro de esta línea de trabajo.

1.3 Descripción del proyecto

La disponibilidad de información de la localización del usuario y de los elementos de su entorno es uno de los elementos clave en el ámbito de la inteligencia ambiental. En entornos exteriores, la tecnología GPS está ampliamente extendida con los sistemas de navegación de vehículos como aplicación más destacada. Por el contrario, en entornos interiores la implantación de los sistemas de localización es escasa y se reduce a aplicaciones específicas, sin que ninguna tecnología concreta se haya impuesto. En particular, la tecnología UWB (Ultra WideBand) es una de las más prometedoras de cara a su uso en sistemas de localización y seguimiento.

Algunos escenarios posibles para un sistema de localización en interiores serían por ejemplo centros comerciales, estaciones de tren, aeropuertos, hospitales, estadios deportivos, etc. Un ejemplo de aplicación para los clientes de un centro comercial sería poderlos situar en un mapa, de manera similar a los sistemas de navegación de los coches, encontrarles la ruta a una determinada tienda, buscarles los restaurantes más próximos y enviarles información de las tiendas cercanas como ofertas o descuentos.

Existen diferentes algoritmos para obtener la posición a partir de las distancias estimadas. Los algoritmos complejos ofrecen una gran precisión, sin embargo requieren un mayor tiempo de computación que algoritmos más sencillos, capaces de calcular la posición consumiendo pocos recursos computacionales, pero con una precisión más limitada. En función de la situación se deberá escoger entre desarrollar dispositivos potentes, capaces de implementar algoritmos complejos o por el contrario desarrollar dispositivos menos potentes que implementen algoritmos menos complejos.

Sin embargo, el corto alcance de los sistemas UWB hace que sean necesarios grandes despliegues si el área de interés es relativamente grande. Además, el posicionamiento requiere unos recursos debido a los intercambios que deben realizarse entre los nodos para estimar las distancias (ranging) y para transmitir las distancias estimadas. Por ello, además de

la precisión del posicionamiento, uno de los objetivos de todo sistema de localización debe ser minimizar la cantidad de recursos (tanto de infraestructura como de comunicación) necesarios.

Una posible mejora de los algoritmos existentes consiste en incorporar información acerca del entorno y de esta forma mejorar su precisión, manteniendo los recursos necesarios o por el contrario, disminuir los recursos consumidos, manteniendo la precisión del sistema. Puede tratarse de información acerca de paredes u obstáculos, o de información estadística acerca de las rutas de los usuarios.

Otra posible solución puede consistir en detectar las situaciones de NLOS, de tal forma, que se trate de corregir la desviación introducida por el obstáculo en la distancia estimada. También se pueden mejorar las prestaciones de los algoritmos realizando un filtrado previo de las estimaciones o mediante una ponderación de las estimaciones en base a su variabilidad.

Para poder evaluar las prestaciones de estos algoritmos avanzados de localización se necesita una herramienta de simulación. Sobre esta herramienta de simulación se analizarán diferentes modificaciones realizadas en los algoritmos de localización.

1.4 Objetivos del proyecto

El principal objetivo de este PFC se centrará en el diseño y evaluación, mediante simulación, de algoritmos avanzados de localización que optimicen el compromiso entre precisión y recursos utilizados mediante el uso de información geográfica y estadística. Para que se pueda utilizar esta información será necesario modificar el simulador ya existente.

Otro de los objetivos del proyecto es la evaluación de distintos algoritmos de localización, y en particular de los propuestos dentro del proyecto EUWB. Para la evaluación de los algoritmos se realizarán simulaciones en diferentes escenarios y situaciones. Además del uso de información geográfica, también se evaluarán otras mejoras a los algoritmos como el prefiltrado y la ponderación de las distancias estimadas en base a su variabilidad.

1.5 Documentación del proyecto

En este apartado se realiza un breve resumen de la documentación presentada en este PFC. La información está compuesta de una memoria y de varios anexos.

- **Memoria:** En este documento se explica la tecnología UWB, los algoritmos utilizados, el simulador empleado, las decisiones tomadas para el diseño del proyecto así como se muestran los resultados y las conclusiones del proyecto. Los puntos principales de la memoria se resumen a continuación:

  Descripción del proyecto: En esta parte de la memoria, se realiza una introducción a los sistemas de localización y a la tecnología UWB. Además se expone el marco del proyecto, una descripción resumida del mismo y se marcan los objetivos a realizar.

  Análisis del sistema de localización: En este apartado se hace una descripción del sistema, de los algoritmos de localización implementados y del simulador. Además se

exponen las mejoras que se van a realizar analizando los requisitos que deben cumplir cada una de ellas.

Diseño del proyecto: Aquí se explican las mejoras realizadas de una forma más detallada, así como se muestran las herramientas utilizadas para el desarrollo de este proyecto. En la última parte, se describe el ciclo de vida y la planificación.

Resultados: En esta parte se muestra un resumen de los resultados obtenidos en las simulaciones. Se muestran gráficas para evaluar las prestaciones de los diferentes algoritmos en términos del error de posicionamiento.

Conclusiones: Se exponen las conclusiones del proyecto. También se proponen posibles mejoras futuras para el proyecto.

Referencias bibliográficas: En este apartado aparecen los libros o documentos que se han utilizado como referencia para realizar este PFC.

- **ANEXO A. Descripción de los algoritmos de localización:** En este anexo se explican al detalle los algoritmos de localización implementados en el simulador.
- **ANEXO B. Descripción del simulador:** En este anexo se realiza una descripción completa del simulador.
- **ANEXO C. Resultados:** En este anexo se analizan la totalidad de los resultados de las simulaciones realizadas.

2. Análisis del sistema de localización

2.1 Descripción del Sistema

El sistema LT (Location & Tracking) considerado tiene como objetivo el seguimiento de la posición de nodos móviles, llamados targets. Se utiliza la tecnología UWB que permite simultanear la localización con la transmisión de información.

Con la finalidad de evaluar diferentes escenarios, diferentes arquitecturas del sistema, diferentes algoritmos de localización y el impacto de diferentes parámetros de diseño, se desarrolló una herramienta de simulación en el marco del proyecto EUWB, utilizando C++ como lenguaje de programación y Visual Studio.NET como plataforma de desarrollo, que en el presente proyecto se ha ampliado en lo referente a algoritmos de localización.

2.1.1 Topología de la red

La red está formada por N_a anchors, que son los nodos fijos y regularmente distribuidos en posiciones conocidas, y N_t targets, que son los nodos móviles que han de ser localizados, como se muestra en la figura 1. Se considera que todo el escenario está cubierto por una única celda UWB, por lo que no se consideran ni la transmisión de datos ni procesos de handover entre diferentes celdas.

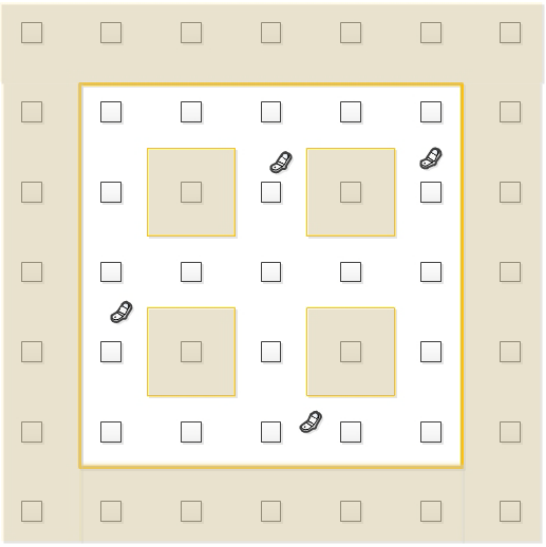


Figura 1. Escenario de simulación

Las distancias estimadas entre los nodos target y los nodos anchor se deben transmitir a un Location Controller (LC), que es la unidad funcional que ejecuta los algoritmos de localización para estimar la posición de los targets y puede estar implementado en uno o varios nodos anchor o en los nodos target.

La topología de la celda es mallada-centralizada (mesh centralized), donde el coordinador transmite tramas piloto (beacon frames) para la sincronización. Después se construye un árbol para la distribución de las tramas piloto y de los comandos desde el

coordinador a cualquier nodo de la red. El árbol se extiende a un árbol mallado (meshed tree) permitiendo la transmisión fuera del árbol de tramas de datos y de ranging, así como las tramas "hello". Las tramas "hello" son enviadas periódicamente por todos los nodos para conocer las relaciones de vecindad.

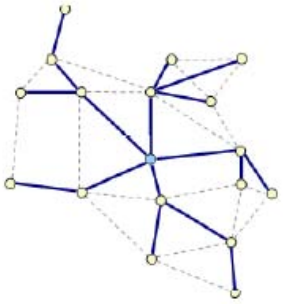


Figura 2. Topología Mesh

2.1.2 Descripción de la capa MAC

Se consideran las capas MAC y PHY diseñadas en el proyecto EUWB y que se basan en el estándar 802.15.4a. La trama MAC está dividida en timeslots, en los que se envían diferentes tipos de tramas (beacon frames, ranging frames, hello frames, data frames y command frames). La estructura de la supertrama es la mostrada en la figura 3.

En cuanto a las tramas de datos, si los nodos origen y destino son vecinos, la trama solo necesitará un timeslot, mientras que si no son vecinos la trama se retransmite a nivel MAC (relaying), necesitándose varios timeslot consecutivos para transmitir un paquete.

Respecto a las tramas de ranging, solo se envían entre vecinos y existen dos tipos: ranging request y ranging response. Ambas utilizan sólo un timeslot.

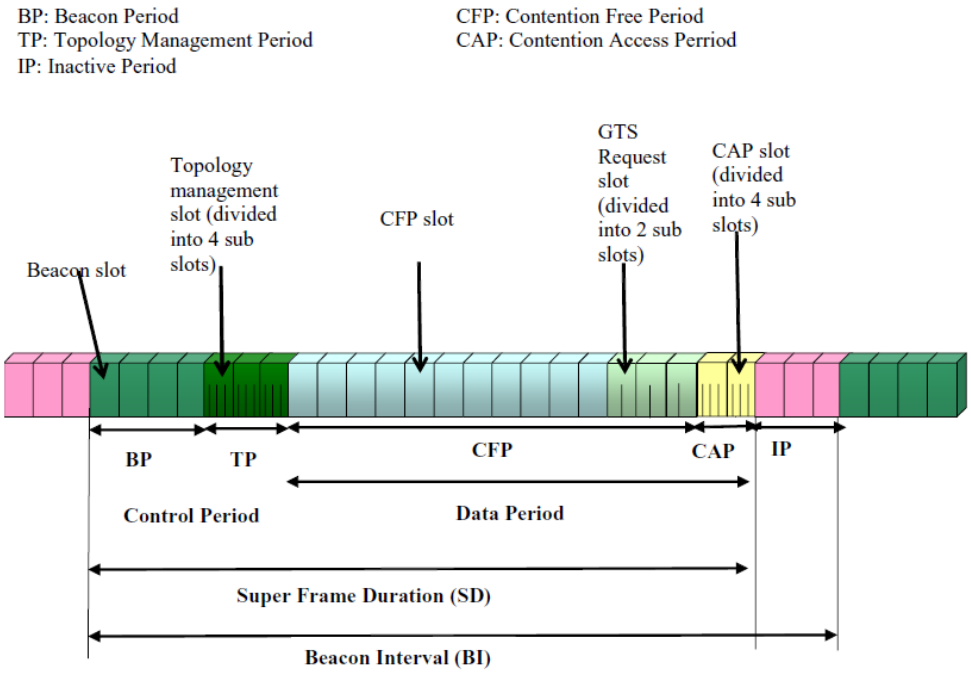


Figura 3. Estructura de la supertrama MAC EUWB

2.1.3 Distribución en la red de la función de tracking

La función de seguimiento o tracking, se lleva a cabo por los Location Controller que pueden estar implementados físicamente en uno o varios anchors o en los targets. En función de donde se implementen, se definen varias arquitecturas:

- Arquitectura centralizada en la red: La función de seguimiento se implementa en uno o varios anchors, que se convierten en LCs.
- Arquitectura centralizada en el target: Son los propios targets los que tienen implementada la funcionalidad de LC y por lo tanto ellos mismos calculan su posición.
- Arquitectura distribuida: Cada target escoge dinámicamente uno de sus anchors vecinos para que realice su seguimiento. Todos los anchors deben tener implementada la funcionalidad de LC.

2.1.4 Adquisición y distribución de la información de localización

La información de localización son las distancias estimadas entre el target y los anchors, que deben ser transmitidas al Location Controller. Según quién inicie el intercambio de ranging, tanto los targets como los anchors pueden estimar la distancia. En el simulador están implementados dos modos para obtener la distancia estimada:

- Two way ranging: El nodo que inicia el proceso, envía un ranging request al otro nodo, que después de un tiempo predefinido envía un ranging response. El iniciador mide el tiempo de llegada de la respuesta y estima la distancia. Se necesitan dos slots.
- Three way ranging: Es similar al Two way ranging, pero en este modo se envían dos ranging responses para compensar la deriva del reloj. El iniciador del proceso es quien estima la distancia. Se necesitan tres slots, pero la estimación es más exacta. Este modo es el que se ha utilizado para realizar las simulaciones de este PFC.

La distancia estimada se calcula a partir de la distancia real, a la que se le añade un error caracterizado por un modelo de ranging. Las medidas de distancia basadas en el tiempo de vuelo, Time of Flight (ToF), estimado a través de transacciones n-Way ranging se pueden modelar de la siguiente forma:

$$\tilde{d}_{ij} = d_{ij} + \varepsilon_{ij} + n_{ij} = d'_{ij} + n_{ij}$$

donde d_{ij} es la distancia real entre los nodos i y j, d'_{ij} es la distancia sesgada, con un sesgo ε_{ij}, y n_{ij} es un término de error residual. La distancia sesgada se modela como la suma ponderada de componentes gaussianas y exponenciales condicionados por la distancia real y la configuración del canal (LOS/NLOS/severe NLOS). De este modo, no solo se tiene en cuenta el efecto multicamino, sino también el efecto de que no haya visión directa entre el target y el anchor. En el modelo estadístico, la probabilidad de que no haya visión directa crece a medida que aumenta la distancia. Mientras que en el modelo basado en paredes y rutas, como se dispone de la distribución de las paredes, se comprueba con los planos si hay o no visión

directa entre el target y el anchor. El error residual se modela como aditivo y centrado, con una varianza σ_n^2 que depende de la resolución en la estimación del tiempo de llegada y de parámetros temporales de los protocolos, y es independiente de la distancia. Tanto el modelo, como los parámetros utilizados han sido obtenidos a partir de estudios de modelado de canal para ranging con dispositivos UWB en entornos interiores, realizados por otros socios del proyecto EUWB.

Una vez que ya se han estimado las distancias, se transmiten al Location Controller en una trama de datos (measurement report data packet). El LC ejecuta el algoritmo de localización para estimar la posición y le transmite esa posición actualizada al target (position update data packet).

2.1.5 Implementación de la función de seguimiento

Con respecto a las técnicas de seguimiento, se puede distinguir entre paramétricas y no paramétricas. Las técnicas paramétricas realizan la localización basándose en un conocimiento a priori del modelo, mientras que las no paramétricas procesan directamente los datos con el uso, en algunos casos, de algunos parámetros estadísticos (media, varianza…). Más adelante se explican con más detalle los diferentes algoritmos de localización que están implementados en el simulador.

Para realizar el seguimiento se puede utilizar la distancia estimada de todos los anchors que se encuentren en cobertura. Sin embargo cuantos más anchors se utilicen, mayor será la cantidad de recursos utilizados. Por eso el simulador permite seleccionar el número de anchors utilizados en el cálculo mediante el fichero de parámetros. Para ello, el simulador dispone de una función para seleccionar los anchors más próximos al target.

2.2 Descripción de los algoritmos de localización

En este punto se hace una breve descripción de los algoritmos de localización. En el Anexo A, se explican detalladamente.

2.2.1 Trilateración

La trilateración es un método matemático para determinar las posiciones relativas de objetos usando la geometría de los triángulos y de las esferas. La trilateración usa las localizaciones conocidas de dos o más puntos de referencia, y la distancia medida entre el sujeto y cada punto de referencia. Para determinar de forma única y precisa la localización relativa de un punto en un plano bidimensional usando sólo trilateración, se necesitan al menos 3 puntos de referencia.

Como hemos comentado, para estimar la posición del target, necesitamos la medida de la distancia de tres anchors diferentes. Si el número de medidas disponibles es inferior a tres, no podemos estimar la posición. Si por el contrario, el número de medidas disponibles es superior a tres, ordenaremos las medidas de tal forma, que nos quedaremos con las medidas de los tres anchors más cercanos, es decir, con las tres medidas de distancia más pequeñas.

2.2.2 Filtro de Kalman

El filtro de Kalman se dedica al problema de intentar estimar el estado $x \in \mathbb{R}^n$ de un proceso discreto en tiempo, gobernado por una ecuación diferencial lineal estocástica. Si la ecuación diferencial no es lineal, se hace referencia al filtro de Kalman Extendido (EKF), que linealiza la ecuación sobre la media actual y covarianza. En el simulador está implementado el filtro de Kalman Extendido. Para la explicación del filtro se ha utilizado la referencia [4].

La ecuación diferencial no lineal con la que trabaja el filtro es:

$$x_k = f(x_{k-1}, u_{k-1}, w_{k-1})$$

Y la medida $z \in \mathbb{R}^m$ es:

$$z_k = h(x_k, v_k)$$

w_k y v_k son variables aleatorias que representan el ruido del proceso y de medida respectivamente. La función no lineal f relaciona el estado en el paso anterior k-1 con el estado en el paso actual k. Incluye como parámetros u_{k-1}, que es una entrada de control opcional, y el ruido del proceso. La función no lineal h relaciona el estado x_k con la medida z_k e incluye como parámetro el ruido de medida.

En la práctica no se conocen los valores individuales del ruido w_k y v_k en cada paso, sin embargo se puede aproximar el estado y la medida sin ellos, de la siguiente manera:

$$\tilde{x}_k = f(\hat{x}_{k-1}, u_{k-1}, 0)$$

$$\tilde{z}_k = h(\tilde{x}_k, 0)$$

Donde $\hat{x}_k$ es la estimación a posteriori del estado. El filtro hace una estimación a priori del estado, que es $\tilde{x}_k$, con el conocimiento del proceso del paso anterior a k. Y posteriormente hace una estimación a posteriori del estado con la medida z_k. Por lo tanto el filtro de Kalman cuenta con dos procesos, el de estimación y el de corrección.

En el simulador el vector de estado incluye la posición p_k, y la velocidad v_k del target. Las ecuaciones de medida utilizadas son de distancias relativas a puntos de posiciones conocidas(anchors) p_i, $i = 1,...,M$. Así que se dispone de M ecuaciones de medida, donde M será el número de anchors utilizados. Las estimaciones del vector de estado y de la medida son:

$$\underbrace{\begin{pmatrix} \tilde{p}_k \\ \tilde{v}_k \end{pmatrix}}_{\tilde{x}_k} = \begin{pmatrix} I & T_s \cdot I \\ 0 & I \end{pmatrix} \underbrace{\begin{pmatrix} \hat{p}_{k-1} \\ \hat{v}_{k-1} \end{pmatrix}}_{\hat{x}_{k-1}}$$

$$\tilde{z}_k(i) = \mid p_i - \tilde{p}_k \mid$$

2.2.3 Filtro de partículas

El filtro de partículas también es conocido como método Monte Carlo secuencial (SMC). Este filtro se utiliza para estimar modelos Bayesianos recursivos. El filtro de partículas se presenta como una alternativa al filtro de Kalman para aplicaciones en tiempo real. En las situaciones donde el modelo no es lineal o el ruido no es Gaussiano, es donde el filtro de partículas tiene más potencial. Se han utilizado las referencias [5] y [6].

Se genera una nube de partículas, cada una con su posición y velocidad, y en cada iteración del filtro se actualizan las partículas, actualizando su posición y su velocidad. A cada partícula se le asocia un peso, que en cada iteración se actualiza en base a la probabilidad de las medidas observadas.

Se consideran sistemas descritos por el modelo genérico de estado:

$$x_{t+1} = Ax_t + B_u u_t + B_f f_t$$

$$y_t = h(x_t) + e_t$$

Donde:

x_t es el vector de estado. Cada muestra del vector de estado se denomina partícula.

u_t son las entradas medidas.

f_t son las fuerzas no medidas o errores.

y_t son las medidas.

e_t es el error de medida.

El modelo de movimiento que se asume es:

$$\begin{pmatrix} p_{t+1} \\ v_{t+1} \end{pmatrix} = \underbrace{\begin{pmatrix} I & T_s \cdot I \\ 0 & I \end{pmatrix}}_{A} \begin{pmatrix} p_t \\ v_t \end{pmatrix} + \underbrace{\begin{pmatrix} T_s^2 / 2 \cdot I \\ T_s \cdot I \end{pmatrix}}_{B_f} f_t$$

Las ecuaciones de medida utilizadas son de distancias relativas a puntos de posiciones conocidas(anchors) p_i , $i = 1,...,M$. Así que se dispone de M ecuaciones de medida, donde M será el número de anchors utilizados:

$$h_i(p_t) = \mid p_i - p_t \mid$$

Este algoritmo consta de una fase de predicción donde se actualiza el estado de las partículas, una fase de actualización de los pesos de las partículas en base a la probabilidad de

las medidas observadas, y una fase de remuestreo donde se reemplazan una fracción de las partículas, para evitar que unas pocas partículas concentren toda la probabilidad.

2.2.4 MDS (MultiDimensional Scaling)

MDS es un método que proporciona una representación espacial de los datos. Dada una matriz que contenga todas las semejanzas-desemejanzas ∂_{ij} entre N puntos, MDS mapea esa configuración en un espacio tal que la diferencia entre ∂_{ij} y las distancias relativas d_{ij} es mínima. Se llama X a la matriz $[N \times \mu]$ que contiene las coordenadas de N objetos en un espacio de μ dimensiones y se escoge el conjunto $\{\partial_{ij}\}$ como el conjunto de distancias euclídeas entre dos objetos i y j del escenario $\{d_{ij}\}$, donde $d_{ij}^2 = (x_i - x_j)^T \cdot (x_i - x_j)$.

De esta forma el problema MDS puede resolverse algebraicamente como la solución de mínimos cuadrados de la matriz $B = X \cdot X^T$. La solución clásica es construir la siguiente matriz de Gram:

$$G = -\frac{1}{2} \cdot J \cdot D^{\circ 2} \cdot J$$

donde $J = I_N - 1_N 1_N^T / N$ es la "centering matrix", que es una matriz simétrica e idempotente, la cual, cuando se multiplica por un vector tiene el mismo efecto que restarle la media de los componentes del vector a cada componente, con 1_N el vector unitario $[N \times 1]$, I_N la matriz identidad $[N \times N]$ y $^{\circ}$ indica el producto elemento por elemento. D es la matriz de distancias euclídeas($[D]_{ij} = d_{ij}$).

La matriz G es equivalente a B (para X centrada en el origen). Por lo tanto, la técnica MDS permite recuperar la localización Y de todos los nodos de la red a partir de la matriz de distancias euclídeas:

$$Y = [V]_{1:\mu} \cdot [\Lambda]_{1:\mu}^{1/2}$$

$$G = V \cdot \Lambda \cdot V^T$$

donde V contiene en sus columnas los vectores propios de G y Λ es una matriz diagonal cuyos elementos de la diagonal son los valores propios de G. Por lo tanto, para poder calcular la matriz Y se debe obtener la matriz de Gram G y después aplicarle la descomposición en valores propios.

La configuración de los nodos que nos devuelve el algoritmo MDS, está mapeada en un sistema de coordenadas diferente al original. Para trasladar las localizaciones Y y obtener las localizaciones reales X, es necesario aplicar una transformación a Y llamada Procrustes. De tal forma que conociendo las coordenadas X_A de al menos $A > \mu$ anchors, la solución Y puede reorientarse mediante esta transformación, obteniendo X. La transformación

Procrustes es una transformación geométrica lineal involucrando únicamente traslación, reflexión, rotación ortogonal y escalado. Se ha utilizado la referencia [7] para la explicación.

2.2.5 Distance Contraction

Como se ha comentado anteriormente al explicar el modelo de ranging, la distancia estimada está sesgada debido al error introducido por el canal (multicamino, NLOS) que es siempre positivo. Una posible estrategia consiste en "contraer" las distancias estimadas para tratar de mitigar el sesgo de la estimación. Mediante el algoritmo distance contraction se contraen las distancias estimadas, que posteriormente se utilizarán en un algoritmo de optimización como por ejemplo SMACOF para hallar la solución óptima. Se ha utilizado la referencia [8] para la explicación del DC.

Se dispone del conjunto de distancias medidas del target a cada anchor i: $\left\{\tilde{d}_i\right\}$. El primer paso del algoritmo es verificar la existencia de la región de viabilidad (feasibility region). Esta región de viabilidad se define como:

$$I \triangleq \left\{\hat{x} \mid \hat{d}_i \le \tilde{d}_i \forall i\right\}$$

donde $\hat{d}_i$ es la distancia del punto $\hat{x}$ a cada anchor i. Si la región de viabilidad no existe, no se puede aplicar el algoritmo de contracción de distancias.

Una vez verificada la existencia de la región de viabilidad, se pueden calcular las distancias contraídas $\overline{d}_i$. Para ello se evalúa la siguiente expresión para cada uno de los anchors:

$$\overline{x}_i = \arg\max_{\hat{x}\in I}(\tilde{d}_i - \hat{d})^2$$

Para cada anchor obtenemos el punto $\overline{x}_i$, que es el punto de tangencia entre la región de viabilidad y la circunferencia con centro la posición del anchor y con radio $\overline{d}_i$, que es la distancia del anchor al punto $\overline{x}_i$. Es necesario conseguir un punto inicial $\hat{x}_0$ dentro de la región, para que después el algoritmo sea capaz de encontrar el punto que maximiza la expresión anterior. Para calcular el punto inicial se utiliza la siguiente expresión:

$$\hat{x}_0 = \arg\min_{\hat{x}\in\mathbb{R}^n} \sum_{i=1}^{N_A} \max\left(0, \hat{d}_i - \tilde{d}_i\right)^2$$

donde N_A es el número de anchors utilizados en la medida.

Finalmente, se sustituyen las distancias medidas $\tilde{d}_i$ por las distancias contraídas $\overline{d}_i$ y se ejecuta un algoritmo de optimización, como por ejemplo SMACOF, para obtener la solución óptima

2.2.6 SMACOF

El algoritmo SMACOF sirve para optimizar la matriz de coordenadas de los anchors y del target, X de dimensiones $[n \times \mu]$, a partir de una solución inicial obtenida mediante otro algoritmo como MDS o distance contraction, de forma que la matriz de distancias derivada de la matriz X sea lo más parecida a la matriz de distancias medidas. Se ha seguido la referencia [9].

Se quiere encontrar la matriz X , tal que $d_{ij}(X) \approx \partial_{ij}$, donde:

$$d_{ij}(X) = \sqrt{\sum_{s=1}^{\mu} (x_{is} - x_{js})^2}$$

El índice $s = 1, ..., \mu$ indica el número de dimensiones del espacio. Se define la función *stress* $\sigma(X)$ de la siguiente forma:

$$\sigma(X) = \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} (\partial_{ij} - d_{ij}(X))^2$$

La matriz W es la matriz $[n \times n]$de pesos w_{ij} , para ponderar las estimaciones en base a su variabilidad. Se asume sin pérdida de generalidad que $\sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} \partial_{ij}^2 = 1$. Para minimizar la función de *stress* se deriva e iguala a cero, obteniendo la configuración X que la minimiza:

$$X = V^{\dagger} B(Y) Y$$

donde los elementos de la matriz $B(Y)$ son $b_{ij} = \begin{cases} -w_{ij} \delta_{ij} d_{ij}^{-1}(Y) & i \neq j \text{ y } d_{ij}(Y) \neq 0 \\ 0 & i \neq j \text{ y } d_{ij}(Y) = 0 \\ -\sum_{k \neq i} b_{ik} & i = j \end{cases}$ y $V^{\dagger}$ es la matriz pseudo-inversa “Moore-Penrose”.

En el simulador se implementa como un procedimiento iterativo, en el paso $t = 0$, se ajusta $Y = X^{(0)}$, donde $X^{(0)}$ es la configuración inicial. En cada iteración se calcula $X^{(t)}$ y después se obtiene $\sigma(X^{(t)})$ y se para de iterar si se cumple la condición: $\sigma(X^{(t)}) - \sigma(X^{(t-1)}) < \varepsilon$ o si se supera un determinado límite de iteraciones.

2.3 Simulador C++

2.3.1 Fichero de entrada: parameters

Parameters.txt es uno de los ficheros de entrada del simulador, mediante el cual configuramos su funcionamiento. Los parámetros más importantes que se pueden configurar se muestran a continuación:

- Modelo del simulador: Statistic o Based on wall & routes plan. El modelo estadístico es el que estaba implementado previamente en el simulador, en el que las distancias estimadas se obtienen en base al modelo estadístico de ranging, de manera independiente en cada iteración y los targets se pueden desplazar en cualquier dirección. El modelo Based on wall & routes plan es el que se va a desarrollar en este proyecto, en el que las distancias estimadas vendrán determinadas por la existencia o no de paredes, de acuerdo al plano definido y el movimiento de los targets vendrá determinado por las rutas definidas.
- Número de targets y velocidad máxima y mínima de los targets.
- Tamaño del area y distancia de separación entre anchors.
- Algoritmos de seguimiento y número de anchors utilizados para el seguimiento.
- Nivel de error residual en la estimación de la distancia (residual ranging noise).

2.3.2 Ficheros de salida: error y resources

Con los ficheros de salida se puede comprobar y evaluar el funcionamiento del simulador, en términos del error de posicionamiento y de los recursos consumidos. En el fichero error.txt se muestran los resultados de la simulación relacionados con el error de posicionamiento. Se muestran tanto los resultados individuales para cada target, como el resultado global. En este fichero se puede observar el número de procesos de posicionamiento y la media, varianza y distribución del error de posicionamiento.

En el otro fichero de salida resources.txt se pueden observar los recursos empleados para la localización, tales como, el número de tramas utilizadas para localización clasificadas en tramas para el envío de distancias medidas, para actualización de la posición, ranging request y ranging response. También se puede observar el tiempo utilizado para localización en segundos y en porcentaje respecto al tiempo total de simulación, así como la tasa de datos usada para localización en bps.

2.3.3 Clase CTarget_node

Esta clase implementa los nodos target a localizar. En esta clase se actualiza la posición real de los targets. En el caso del modelo estadístico, la dinámica de los targets se modela por direcciones y velocidades aleatorias que son constantes durante un periodo de tiempo determinado, después del cual se calcula una nueva velocidad y dirección. El modelo dinámico se define mediante unas velocidades máxima y mínima y la tasa de cambio de dirección.

En esta clase se implementan también la función para estimar las distancias a cada anchor y las funciones que recogen las estadísticas del error de posicionamiento.

2.3.4 Clase CAnchor_node

Esta clase define los nodos anchor, que son los nodos con posiciones fijas y conocidas. Los anchors se encargan de transmitir la información de localización de los target al location manager y sirven de referencia para el posicionamiento de los targets.

2.3.5 Clase CLocation_manager

Este elemento es el encargado de estimar la posición de los target, a partir de las distancias estimadas a los anchors. Puede estar implementado en los target o en los anchors, en función de la estrategia seleccionada. Para estimar la posición necesita conocer las posiciones de los anchors, las distancias estimadas, la última posición estimada del target y el algoritmo que se ha de utilizar.

Esta clase también implementa la función que selecciona los anchors que se van a utilizar en la medida y la función que indica cuando es necesario actualizar la posición del target.

2.3.6 Clase CPicocell_manager

El picocell manager controla la transmisión de tramas y el uso de los slots. Se encarga de los procesos de ranging, de transmisión de información y de actualización de las relaciones de vecindad entre los targets y los anchors. Están implementadas las colas de transmisión y de ranging, que almacenan las solicitudes de transmisión y de ranging respectivamente, para ser procesadas posteriormente.

Este elemento se encarga también de recoger las estadísticas relacionadas con los recursos consumidos.

2.3.7 Clase CNetwork

Esta clase es la encargada de implementar la topología de la red, incluyendo los elementos: nodos anchor, nodos target, location manager y picocell manager. También realiza la carga de los parámetros, y en el caso del modelo de simulador basado en paredes y rutas, también se encarga de cargar los diferentes escenarios y las probabilidades de los targets de seguir recto, girar o darse la vuelta cuando llegan a un nodo del plano. Además, realiza la distribución geográfica de los nodos anchor y de los location managers, la asignación de los targets a cada location manager y el cálculo del número de saltos de cada nodo anchor al location manager más cercano.

2.3.8 Implementación de los algoritmos de localización

Trilateración

El algoritmo de trilateración está implementado en la clase CLocation_manager, dentro de la función: Calculate_position. En diferentes variables estarán almacenadas las posiciones de los tres anchors escogidos, así como las tres distancias medidas, para poder desarrollar el algoritmo y obtener la posición del target.

Filtro de Kalman

El Filtro de Kalman se ha implementado utilizando la librería kfilter. El filtro está desarrollado en la clase CPlane_EKF. Esta clase dispone de las funciones: makeA, makeH, makeV, makeR, makeW y makeQ para construir las matrices A, H, V, R, W y Q. También dispone de las funciones makeProcess, en la que se almacena en un vector temporal el nuevo vector de estado, y makeMeasure, donde se actualiza el vector de medidas.

La posición y la velocidad inicial del target se obtienen mediante trilateración. Se ejecuta la trilateración las dos primeras veces, para así obtener dos posiciones consecutivas del target y poder hacer una estimación de la velocidad inicial del target. En la clase CLocation_manager, dentro de la función Calculate_position, se inicializa el filtro de Kalman y en cada estimación se llama a la función Step, que ejecuta el filtro de Kalman.

Filtro de Partículas

El filtro de partículas se ha desarrollado utilizando la librería smctc [10]. El filtro está implementado en el fichero fuente pffuncs.cc. Para desarrollar el filtro, este fichero incluye la clase cv_state, que almacena la posición y velocidad, tanto en el eje x como en el eje y, para cada partícula. También existe la clase cv_obs, para almacenar la posición y la distancia medida de cada anchor.

Las funciones realizadas por el fichero son:

- fInitialise: Inicializa las partículas, obteniendo una posición y una velocidad aleatorias con distribución normal, de media la posición inicial del target y la velocidad inicial del target respectivamente. La posición inicial y la velocidad inicial se obtienen mediante trilateración, de la misma forma que en el filtro de Kalman.

- fMove: Obtiene una aceleración para el eje x y otra para el eje y de forma aleatoria y actualiza las particulas, actualizando la posición(con la velocidad y la aceleración) y la velocidad(con la aceleración).

- logLikelihood: Calcula el logaritmo de la probabilidad para todas las partículas. Se dispone de tres modelos para calcularla, basados en una gaussiana, en la suma de dos gaussianas y en la suma de tres gaussianas respectivamente. Para calcular la probabilidad se tienen en cuenta las posiciones de la particula, de los anchors y las distancias medidas.

- integrand_mean_x: Calcula la posición en el eje x del target, a partir de las partículas.

- integrand_mean_y: Calcula la posición en el eje y del target, a partir de las partículas.

En la clase CLocation_manager, dentro de la función Calculate_position, se inicializa el filtro de partículas y en cada estimación se llama a la función Iterate, que ejecuta el filtro de partículas.

WLS-MDS (propuesta PULSERS PHASE II)

Este algoritmo es el propuesto en el marco del proyecto PULSERS PHASE II [11] y se basa en el uso de MDS para obtener una estimación inicial de la posición y en SMACOF para la optimización de la solución. El WLS-MDS se ha adaptado del desarrollo en C# realizado por otro socio dentro del proyecto EUWB y está implementado en la clase CMDSalgorithm. Dentro de esta clase, la función Centralized_Algorithm es la que genera la matriz de distancias, a partir de las distancias entre anchors y de las distancias medidas, y la matriz de pesos para ponderar las estimaciones en base a su variabilidad, mediante la función LINK_Weight. Una vez obtenidas ambas matrices, se llama a la función MDSMAP, que lleva a cabo el algoritmo MDS. Con este algoritmo obtenemos las posiciones del target y de los anchors que mejor se ajustan a la matriz de distancias calculada previamente.

La solución obtenida del MDS se puede optimizar mediante el algoritmo SMACOF, implementado por la función smacofEUWBW. Se utiliza la librería Altaxo para obtener la matriz pseudo-inversa necesaria para el algoritmo. Para obtener la posición válida del target, se debe reorientar la solución mediante el algoritmo Procrustes. Para realizar las operaciones con matrices se ha utilizado la clase CGeneralMatrix, que se ha adaptado del desarrollo en C# realizado en el proyecto EUWB. La transformación Procrustes tambien se ha adaptado del desarrollo en C# y está desarrollada en la clase CProcrustes.

WLS-DC (propuesta EUWB)

Este algoritmo es el propuesto en el marco del proyecto EUWB y se basa en el uso de MDS para obtener una estimación inicial de la posición, distance contraction para contraer las distancias estimadas y SMACOF para la optimización de la solución. Está implementado en la clase CMDSalgorithm. Como en el caso anterior, este algoritmo se ha adaptado del desarrollado en C# por otro socio dentro del proyecto EUWB. La función Centralized_Algorithm_EUWB genera la matriz de distancias y la matriz de pesos de la misma forma que la función Centralized_Algorithm, y llama a la función MDSMAP, para realizar el MDS.

La solución obtenida del MDS se optimiza mediante los algoritmos Distance Contraction y SMACOF. Con el algoritmo Distance Contraction, implementado en la función DistContrPos, se obtiene la matriz de distancias contraidas. Las funciones de minimización y maximización que se realizan dentro del algoritmo Distance Contraction se realizan mediante la librería DotNumerics. Con esa matriz de distancias, y con las posiciones del target y de los anchors obtenidas en el MDS, se ejecuta el algoritmo SMACOF. Para obtener la posición válida del target, se reorienta la solución mediante el algoritmo Procrustes, desarrollado por la clase CProcrustes.

2.4 Requisitos del sistema

En este apartado se analizan los requisitos que tiene que cumplir el proyecto y las mejoras que se deben implementar en el simulador C++ ya existente. Debido a que se parte de

un simulador previo, el principal requisito consiste en que todos los cambios introducidos deben ser compatibles con el simulador previo.

2.4.1 Adaptación a escenario basado en paredes

Esta primera modificación consiste fundamentalmente en modelar los escenarios, modificar el movimiento de los targets para que se adapten a esos escenarios y modificar la estimación de las distancias.

Modelado de escenarios

Esta modificación consiste en diseñar una serie de escenarios de simulación. Los escenarios se compondrán de un listado de paredes, así como de las posibles rutas de los móviles definidas por nodos (intersecciones) y segmentos (pasillos). El principal requisito de esta modificación es conseguir cargar la información del escenario en el simulador y que esa información esté disponible para los algoritmos de localización.

Movimiento de los targets

La siguiente mejora consiste en modificar el movimiento de los target, para que se ajuste a los pasillos del escenario cargado. El target se desplazara de un nodo a otro del escenario siguiendo los segmentos definidos en el plano. Se escogerá un nodo destino aleatoriamente en base a las probabilidades definidas para el nodo actual y se escogerá una velocidad aleatoria que se mantendrá constante hasta que el target alcance el nodo destino.

Un requisito de esta modificación es permitir que el modelo de movimiento anterior y el nuevo puedan coexistir, de tal forma que según qué modelo de simulador escojamos, los targets se muevan de una forma u otra.

Estimación de las distancias

En el simulador existente, la estimación de las distancias se realiza en base al modelo estadístico de ranging, de manera independiente en cada iteración. Se modificará la estimación de las distancias, de tal forma que se calculará el número de paredes existentes entre el anchor y el target de acuerdo al plano definido. Si no existen paredes entre el anchor y el target se estará en condiciones de LOS. Y si existen paredes se estará en condiciones de NLOS.

El principal requisito de esta modificación es poder calcular el número de paredes existentes entre el target y cada anchor utilizado.

2.4.2 Implementación de algoritmos avanzados

En esta segunda modificación se adaptarán los algoritmos WLS-MDS y WLS-DC, propuestos en PULSERS PHASE II y EUWB respectivamente, al simulador C++ existente. Además se implementarán un prefiltrado y una ponderación de las distancias estimadas.

Adaptación de los algoritmos propuestos en PULSERS PHASE II (WLS-MDS) y EUWB (WLS-DC)

Esta modificación consiste en adaptar al simulador existente en C++ los algoritmos WLS-MDS y WLS-DC del desarrollo en C# realizado por otro socio dentro del proyecto EUWB. Además de adaptar estos algoritmos, se comprobará su correcto funcionamiento y se optimizarán algunos parámetros.

Prefiltrado de las medidas

La mejora del prefiltrado consiste en filtrar las distancias estimadas, antes de pasárselas al algoritmo de localización. En el simulador se implementara un filtro paso bajo, realizado mediante una ponderación de la distancia estimada actual y las distancias estimadas anteriores. Después esa nueva distancia se envía al algoritmo de localización para que estime la posición del target.

Un requisito del prefiltrado es que nos permita elegir con cuantas muestras queremos realizar la ponderación, y nos permita elegir también cuales son los pesos de cada distancia en la ponderación. Si elegimos una única muestra para el prefiltrado, se utilizará la distancia estimada actual sin ningún prefiltrado.

Otro requisito de esta modificación es que necesitamos almacenar las distancias estimadas anteriores, para poder realizar la ponderación. En concreto se almacenan la distancia actual y las cuatro distancias anteriores, con lo que siempre se tienen cinco distancias estimadas para realizar el prefiltrado.

Ponderación de las medidas

En esta modificación se trata de implementar en los algoritmos de localización una ponderación de las distancias medidas en base a su variabilidad, de tal forma que aquellas medidas cuya varianza sea elevada tendrán poco peso en la ponderación. Mientras que las que tengan poca varianza, serán las que tengan mayor peso.

El requisito más importante de esta modificación es que se necesitan almacenar las distancias estimadas anteriores, para poder obtener la varianza de la medida, y poder obtener los pesos de la ponderación.

2.4.3 Utilización de información geográfica

Esta mejora consiste en modificar alguno de los algoritmos de localización presentes en el simulador, para que utilicen información geográfica de los planos del escenario y de esta forma optimizar el compromiso entre precisión y recursos utilizados. Fundamentalmente esta modificación consta de dos bloques que son: Identificar situaciones NLOS y utilizar información acerca de las rutas de los targets. Un requisito de esta mejora es añadir estas modificaciones a los algoritmos, manteniendo la posibilidad de utilizar las versiones no mejoradas de los algoritmos. A través del fichero de parámetros es donde se selecciona el algoritmo de localización deseado.

Identificación de situaciones NLOS

Esta modificación consiste fundamentalmente en identificar situaciones de NLOS e utilizar dicha información para mejorar las prestaciones del algoritmo, bien a través del modelo de las observaciones en el caso de los algoritmos paramétricos, o a través de la ponderación en el caso de los algoritmos WLS-MDS y WLS-DC.

Para identificar las situaciones NLOS se requiere detectar la existencia de paredes entre el target y los anchors.

Utilización de información acerca de las rutas de los targets

Con esta mejora se pretende que los algoritmos utilicen información sobre las rutas que suelen seguir los targets, para así mejorar la precisión de la estimación de la posición o para disminuir los recursos necesarios. Esta mejora será aplicable a los algoritmos que utilizan el modelo dinámico de los targets, concretamente el filtro de Kalman y el filtro de partículas.

Se requiere un modelado de las rutas que siguen los targets en los planos, para poder extraer de ahí la información. En base a la información disponible de las rutas, se modificará el modelo dinámico utilizado en los algoritmos.

3. Diseño del proyecto

3.1 Descripción del diseño

En este apartado se describen las mejoras implementadas con más detalle.

3.1.1 Adaptación a escenario basado en paredes

Modelado de escenarios

Esta modificación consiste en modelar y cargar los escenarios en el simulador. Se han modelado tres escenarios diferentes, para tres distancias entre anchors diferentes. Los tres escenarios se muestran en las figuras 4, 5 y 6.

Para que los targets se muevan dentro de los escenarios, se define como nodo la intersección de dos pasillos y como segmento el pasillo que une dos nodos. Las características de los escenarios se cargan a través de los ficheros plan.txt. Cuando se selecciona la distancia entre anchors en el fichero de parámetros, el simulador selecciona automáticamente el fichero plan.txt correspondiente a esa distancia. En estos ficheros se introduce la información de las paredes, los nodos y los segmentos:

- Paredes: En el caso de las paredes se introducen las coordenadas de los dos extremos de la pared (las paredes implementadas en el simulador sólo pueden ser rectas).
- Nodos: Para los nodos se introducen sus coordenadas y los identificadores de sus nodos vecinos.
- Segmentos: En los segmentos se indican los identificadores de los dos nodos que une.

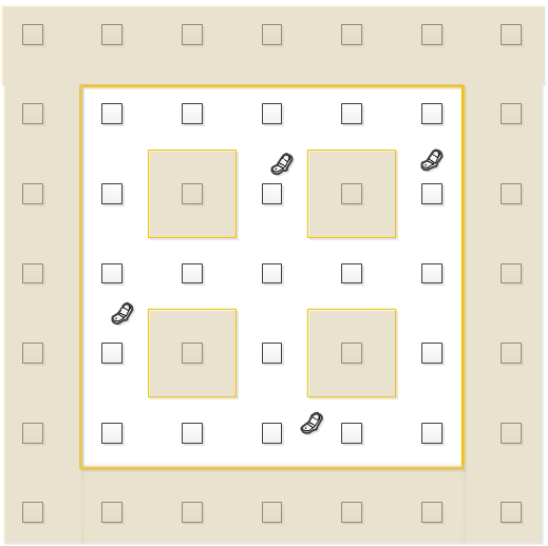


Figura 4. Escenario para una distancia entre anchors de 8.33 m

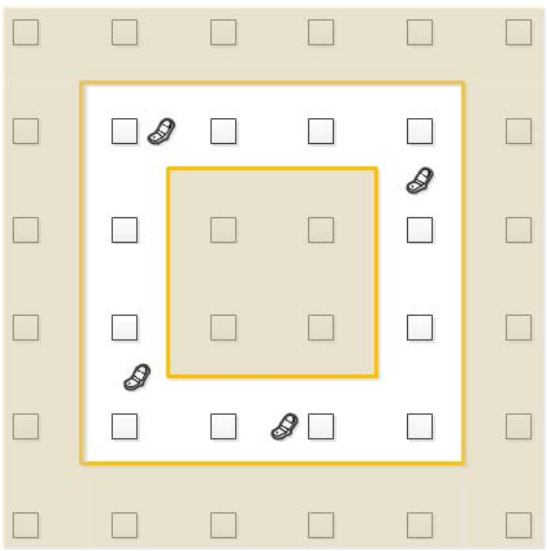


Figura 5. Escenario para una distancia entre anchors de 10 m

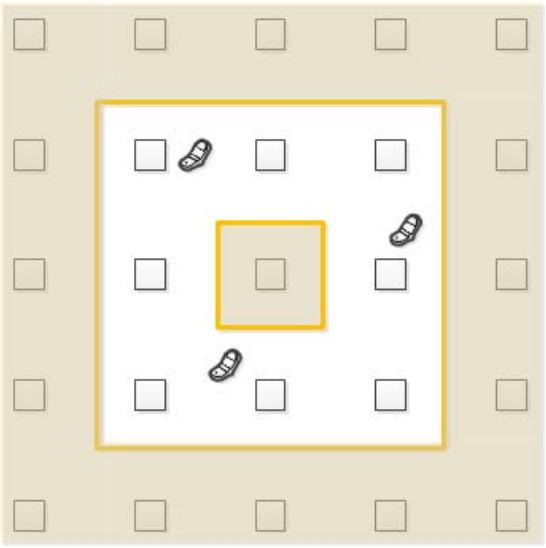


Figura 6. Escenario para una distancia entre anchors de 12.5 m

Para cargar los datos de los ficheros plan.txt se ha implementado la función Load_plan, dentro de la clase CNetwork. Esta función lee los datos del fichero y almacena la información en tres vectores. Uno de ellos contendra la información asociada a las paredes, otro a los nodos y otro a los segmentos.

Movimiento de los targets

La modificación consiste en cambiar el movimiento de los targets. El modelo anterior, modelo estadístico del simulador, consistía en elegir una dirección y una velocidad aleatorias, que se mantenían constantes durante un determinado tiempo, y actualizar la posición en cada paso. Con la modificación se quiere que los targets se muevan dentro de un determinado escenario.

Como se ha comentado en la modificación anterior, se llama nodo, a la zona de intersección de dos pasillos y segmento al pasillo que une dos nodos. En el escenario mostrado en la figura 4, en concreto hay 9 nodos y 12 segmentos.

Para implementar esta mejora se ha modificado la función Update_real_position dentro de la clase CTarget_node. Cuando se inicializa un target, se escoge aleatoriamente un nodo inicial. Del plano cargado disponemos de las coordenadas de los nodos, que es la posición central de la zona donde se unen dos pasillos. Para determinar la posición inicial del target, se le suma a las coordenadas del nodo, un desplazamiento aleatorio en los dos ejes, que sigue una distribución uniforme entre -1 y 1. Una vez obtenida la posición inicial se escoge aleatoriamente uno de los nodos vecinos, y de nuevo para obtener la posición final del target se les suma un desplazamiento uniforme a las coordenadas del nodo. Se obtiene una velocidad aleatoria y se calcula el desplazamiento que hay que realizar en cada iteración en ambos ejes, con la velocidad y con el tiempo entre iteración e iteración. En cada paso se actualiza la posición real del target sumándole a la posición anterior los desplazamientos en ambos ejes.

Cada vez que el target llega al nodo destino, este pasa a ser el nodo inicial y se escoge uno de sus nodos vecinos como nodo destino, según las probabilidades contenidas en un fichero de texto, likelihood.txt. En este fichero se encuentran las probabilidades de que cuando un target llega al nodo destino siga recto hasta el nodo siguiente, se dé la vuelta y vuelva al mismo nodo, o gire a izquierda o derecha.

Una vez escogido el nodo destino, se obtiene de nuevo una posición final, como la posición del nodo más un desplazamiento uniforme, y una velocidad, y se calculan los desplazamientos que hay que realizar en cada paso, para actualizar la posición del target. De esta forma el target se desplaza por los pasillos de los escenarios del simulador.

Estimación de las distancias

Para esta modificación hay que cambiar la estimación de las distancias, que se realiza en la función Estimate_distance, dentro de la clase CTarget_node. En el simulador existente, la estimación de las distancias se realiza en base al modelo estadístico de ranging de manera independiente en cada iteración. En función de la distancia, se calcula la probabilidad de que el enlace entre target y anchor sea LOS, NLOS o NLOS severo y se calcula el error sumando las tres componentes ponderadas por su probabilidad correspondiente. Para el nuevo modelo, es necesario calcular el número de paredes existentes entre el target y los anchors utilizados en la medida, y en base al número de paredes se determina la componente de error que se utiliza.

Para calcular el número de paredes, se ha implementado una función llamada n_cutoff_point, que calcula el número de puntos de corte entre el rayo directo que une el target y el anchor y las paredes del escenario. Si no existe ninguna pared, el enlace entre el target y ese anchor se encuentra en condiciones de LOS. SI existe una pared, el enlace está en situación de NLOS. Y si existe más de una pared, el enlace se encuentra en condiciones severas de NLOS.

3.1.2 Implementación de algoritmos avanzados

Adaptación de los algoritmos propuestos en PULSERS PHASE II (WLS-MDS) y EUWB (WLS-DC)

En esta modificación se adaptaron los algoritmos WLS-MDS y WLS-DC, propuestos en PULSERS PHASE II y EUWB respectivamente, del desarrollado en C# por otro socio dentro del proyecto EUWB. Se tradujeron ambos algoritmos a C++ y se modificaron para adecuarlos al simulador existente.

El algoritmo WLS-MDS se adaptó en la función Centralized_Algorithm, dentro de la clase CMDSalgorithm. Y el algoritmo WLS-DC se adaptó en la clase CMDSalgorithm, en la función Centralized_Algorithm_EUWB.

Prefiltrado de las medidas

Para realizar esta mejora es necesario almacenar las cuatro distancias estimadas anteriores, para tener un total de cinco muestras para realizar la ponderación. Se puede elegir

el número de muestras a ponderar y los pesos de la ponderación. Si se escoge una sola muestra, no se realizará el prefiltrado. La finalidad del prefiltrado es reducir la variabilidad de las muestras provocada por el error residual.

Se ha implementado una función llamada Update_target_to_anchor_distances, dentro de la clase CLocation_manager, que se llama cada vez que se estima una nueva distancia y es la encargada de eliminar la distancia almacenada más vieja y añadir la nueva.

El prefiltrado se ha implementado para el algoritmo LS-MDS, de tal forma que cuando se calcula la matriz de distancias, en lugar de introducir la distancia estimada actual entre el target y cada anchor, se introduce la ponderación de las últimas distancias. De esta forma se realiza el prefiltrado.

Ponderación de las medidas

Para la ponderación se ha adaptado una función llamada LINK_Weight, del desarrollo en C# realizado por otro socio dentro del proyecto EUWB, que calcula una matriz de pesos, en función de la varianza de las distancias estimadas, de tal forma, que cuanto mayor sea la varianza de las medidas menor será el valor del peso. Con esto se consigue dar más valor a los enlaces cuya variabilidad es menor, mientras que los enlaces con mayor variabilidad, que se supone que son aquellos que se encuentran en situación de NLOS, tienen menos valor.

Los pesos pueden tomar valores entre cero y uno. Para calcular la varianza de las distancias estimadas utilizamos las cinco distancias estimadas almacenadas, cuyo almacenamiento se implementó en la mejora del prefiltrado. Posteriormente esa matriz de pesos se utiliza en la optimización mediante el algoritmo SMACOF, y en el caso del algoritmo WLS-DC, también se utiliza en el algoritmo Distance Contraction.

3.1.3 Utilización de información geográfica

Identificación de situaciones NLOS

Esta mejora consiste en utilizar la información disponible acerca de la situación de las paredes para identificar situaciones de NLOS, e utilizar esta información para la mejora de los algoritmos. La identificación de situaciones NLOS se ha desarrollado en WLS-MDS, WLS-DC, filtro de Kalman y filtro de partículas.

Para la identificación de situaciones NLOS se utiliza la función que se implementó en la modificación de la estimación de las distancias, que se llama n_cutoff_point y que a través de la posición del target y de los anchors, calcula si el rayo directo entre el target y el anchor cruza alguna pared. Esta función devuelve el número de paredes que cruza el rayo directo. De esta forma se sabe si un enlace está en condiciones de NLOS.

Para el filtro de partículas la mejora consiste en comprobar para cada medida entre target y anchor si se encuentra en situación de LOS o de NLOS. A cada partícula se le asigna una probabilidad, que es el producto de las probabilidades de la partícula en base a la distancia estimada con cada uno de los anchors utilizados en la medida. Las probabilidades de la

distancia estimada a cada anchor se calculan como la suma ponderada de dos o de tres gaussianas, según el modelo que se escoja. En el caso de no detectar situaciones de NLOS, los pesos LOS, NLOS y NLOS2 (severe NLOS), se calculan en función de la distancia entre la partícula y el anchor, de tal forma que cuanto más lejos este la partícula, más valor tomaran los pesos NLOS y NLOS2. En el caso de detectar los cruces con las paredes, si hay situación de LOS, su peso toma el valor uno y los restantes toman el valor cero. En el caso de cruzar una pared, el que toma valor uno es el peso NLOS. Y si hay dos o más cruces, toma valor uno el peso NLOS2.

En el filtro de Kalman la modificación se basa en detectar, con la posición estimada del target y las posiciones de los anchors, si se está en situación de LOS o de NLOS para cada anchor. Cuando se construye la matriz R, que es la matriz de covarianza del ruido de medida, si se está en situación de LOS se introduce como valor de covarianza 0.7; mientras que en situaciones de NLOS se introduce como valor de covarianza 1.5x0.7. De esta forma se consigue que los enlaces en NLOS tengan una varianza mayor. Tras diversas pruebas se vio que el factor 1.5 era el que mejor resultados ofrecía.

Para WLS-MDS y WLS-DC, las distancias estimadas se ponderaban en función de su varianza. En el caso de utilizar identificación de NLOS, se comprueba sí el target está en situación de LOS o de NLOS para cada anchor y si está en NLOS, se multiplica el peso obtenido por la función LINK_Weight por 0.5, con lo que los enlaces en NLOS tienen menor peso. El factor 0.5 se determinó mediante simulaciones.

Utilización de información acerca de las rutas de los targets

Con esta mejora se modifican el filtro de Kalman y el filtro de partículas, para que utilicen información sobre las rutas que siguen los targets y así mejorar la precisión del sistema o reducir los recursos necesarios para localización.

Para el filtro de partículas se han implementado tres funciones; una de ellas se llama check_particle y comprueba sí una partícula esta dentro de las rutas posibles o sí está fuera. La segunda función se llama check_position, y comprueba si el target se encuentra en un segmento del escenario, o si por el contrario se encuentra en un nodo. La tercera función se llama change_direction y se utiliza para modificar la dirección de las partículas.

La primera función se utiliza a la hora de calcular la probabilidad de cada partícula, de tal forma que si la partícula está fuera de las rutas posibles, se divide su probabilidad por un determinado factor y así se consigue disminuir la probabilidad de las partículas que se encuentren fuera de las rutas posibles del escenario. En un primer momento se pensó en eliminar las partículas fuera de las rutas posibles, bien dándole peso cero o bien forzando mediante el modelo dinámico que las partículas estén siempre dentro de las rutas posibles. Sin embargo, esto empeoró los resultados, ya que se perdía en gran medida la diversidad y aleatoriedad del filtro. Tras diversas pruebas, se estimó que dividir el peso de las partículas fuera de las rutas por un factor 6 era la opción que presentaba los mejores resultados.

La segunda función se utiliza para modificar el movimiento de las partículas cuando el target se encuentra en un nodo. Por el contrario, cuando el target se encuentra en un segmento no se modifica nada. Cuando se detecta que el target está en un nodo, se modificará la dirección de un determinado porcentaje de las partículas. De esta forma, dentro del grupo de partículas a modificar, la dirección se modificará de acuerdo a las probabilidades determinadas en el fichero likelihood para seguir recto, girar a la derecha, girar a la izquierda o dar media vuelta. La dirección de movimiento de las partículas se cambia con la función change_direction. El porcentaje de partículas cuya dirección se modifica se determinó tras diversas pruebas y se fijo en un 80%, ya que era el que mejores resultados proporcionó. También se realizaron multiples pruebas cuando el target está en un segmento, tratando de identificar la dirección del target y forzando las direcciones de las partículas en ese sentido. Sin embargo los resultados obtenidos fueron peores, por lo que finalmente no se implementaron.

Para el filtro de Kalman se ha modificado la función que calcula la matriz Q. En cada iteración del filtro se comprueba si el target se encuentra en un nodo o si se encuentra en un segmento. Si se encuentra en un segmento no se modifica nada, se introduce la varianza de la aceleración que está por defecto que es 3 m/s^2, pero si se encuentra en un nodo, se multiplica este valor por 5, introduciendo por lo tanto una varianza de 15 m/s^2. El factor 5, se determinó tras varias simulaciones. Esto se realiza únicamente en los nodos, ya que es cuando el target puede realizar un cambio brusco de dirección, por ello se aumenta la varianza de la aceleración, para que el filtro pueda responder mejor a ese posible cambio brusco de dirección.

3.2 Herramientas utilizadas

Para la realización de este PFC, se han utilizado las siguientes herramientas:

- C ++ como lenguaje de programación.
- Visual Studio.NET como plataforma de desarrollo.
- Microsoft Excel para el tratamiento de los datos.
- Microsoft Visio para los gráficos y para los diagramas de Gantt.
- Microsoft Word para la redacción de la memoria y otra documentación

3.3 Proceso de desarrollo

3.3.1 Ciclo de vida

El ciclo de vida de este proyecto se ha basado en el ciclo de vida iterativo e incremental, basado en el modelo en espiral de Barry Boehm, ya que era el más adecuado para este proyecto. Este modelo consiste básicamente en una serie de ciclos que se repiten en forma de espiral, comenzando desde el centro. En la figura 7 se muestra un diagrama de este modelo.

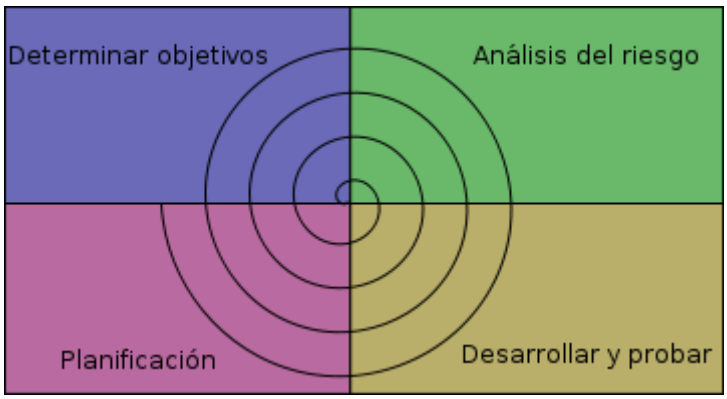


Figura 7. Diagrama modelo en espiral

Las cuatro fases principales en este tipo de ciclo de vida son:

- Determinar objetivos: En esta fase se fijan los objetivos, las alternativas y las restricciones del proyecto.
- Análisis de riesgos: En la segunda etapa, se identifican los riesgos del proyecto y las estrategias alternativas para resolverlos.
- Desarrollar (Ingeniería): En esta tercera fase, se desarrolla el proyecto.
- Evaluación y Planificación: En la cuarta fase se revisa y evalúa todo lo realizado, y con ello se decide sí se continúa y se planifica la siguiente actividad.

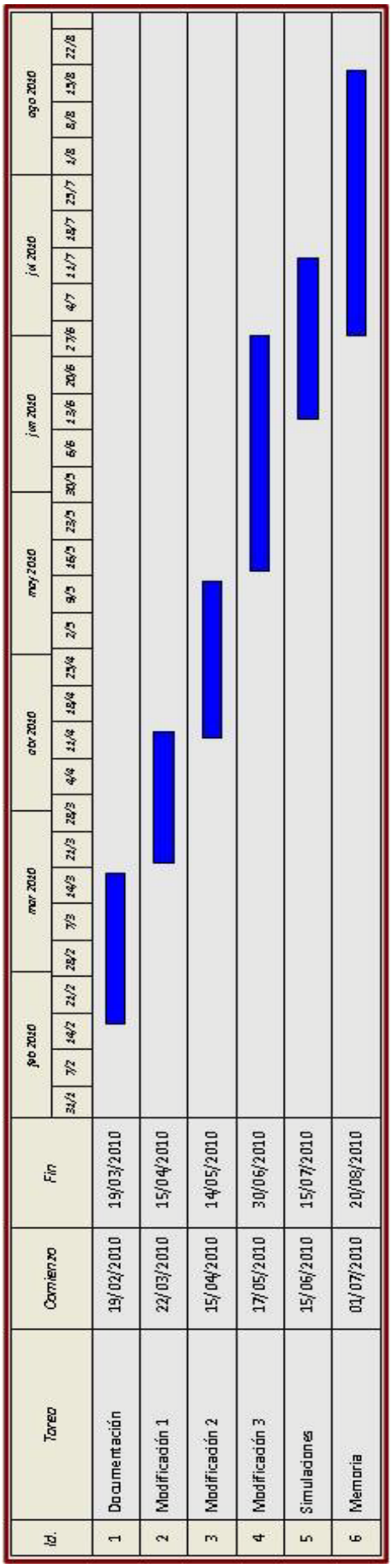
3.3.2 Planificación

En este apartado se presenta la planificación de las tareas del proyecto, en la que se estiman las horas a invertir en cada tarea. La planificación es el primer paso en un proyecto, ya que esta planificación aborda desde el inicio hasta la finalización del proyecto.

Se realizó una planificación inicial y en este punto vamos a compararla con la planificación final, que son las horas invertidas realmente en cada tarea. De esta forma se podrán analizar las posibles causas de los desfases entre la planificación inicial y la planificación real o final.

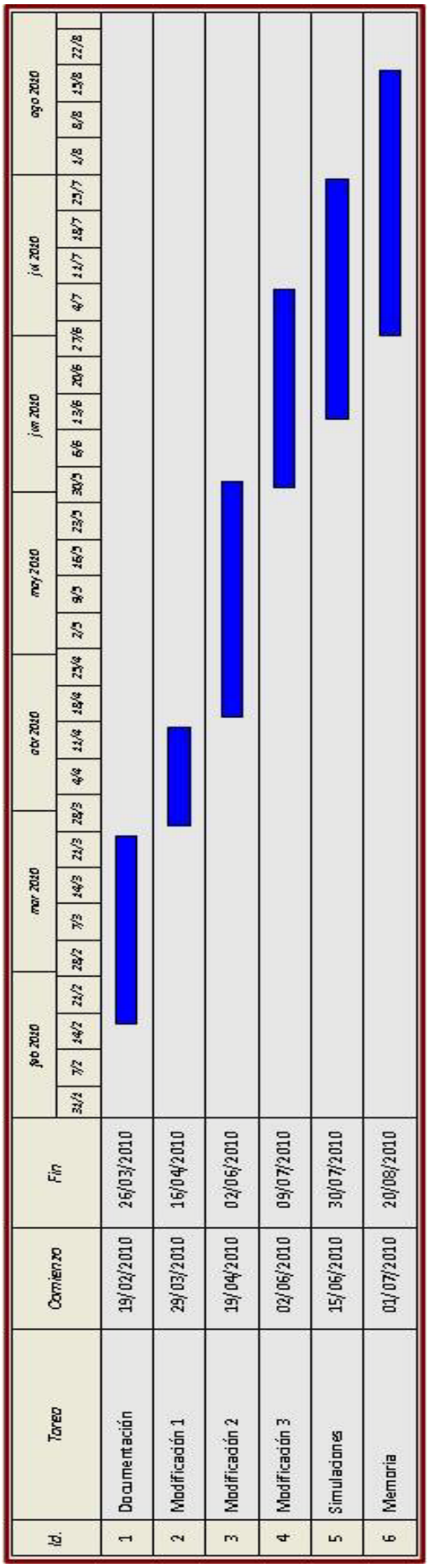
Planificación inicial

En la figura 8 se muestra el diagrama de Gantt de la planificación inicial del proyecto. Más adelante se discutirán las diferencias con la planificación realizada al finalizar el proyecto.



Id.	Tarea	Comienzo	Fin
1	Documentación	19/02/2010	19/03/2010
2	Modificación 1	22/03/2010	15/04/2010
3	Modificación 2	15/04/2010	14/05/2010
4	Modificación 3	17/05/2010	30/06/2010
5	Simulaciones	15/06/2010	15/07/2010
6	Memoria	01/07/2010	20/08/2010

Figura 8. Diagrama de Gantt planificación inicial



Id.	Tarea	Comienzo	Fin
1	Documentación	19/02/2010	26/03/2010
2	Modificación 1	29/03/2010	16/04/2010
3	Modificación 2	19/04/2010	02/06/2010
4	Modificación 3	02/06/2010	09/07/2010
5	Simulaciones	15/06/2010	30/07/2010
6	Memoria	01/07/2010	20/08/2010

Figura 9. Diagrama de Gantt planificación final

Hitos del proyecto

Se propusieron los siguientes hitos para este proyecto:

Modificación 1	22 de Marzo de 2010
Modificación 2	15 de Abril de 2010
Modificación 3	17 de Mayo de 2010
Simulaciones	15 de Junio de 2010

Planificación final

El diagrama de Gantt de la planificación final del proyecto se muestra en la figura 9. En la planificación inicial no se contemplaba la adaptación del algoritmo WLS-DC, aunque mientras se realizaba la adaptación del WLS-MDS se vio que podría ser interesante implementarlo.

Desajuste entre planificación inicial y final

Observando ambas planificaciones se puede observar que el primer desajuste se produjo en la etapa de Documentación que duró algo más de lo esperado, ya que tenía que aprender a programar un lenguaje nuevo para mí; Además también tenía que entender la estructura del simulador ya existente. Para el aprendizaje de C++ utilice la referencia [12].

La primera modificación, se realizó en menos tiempo del esperado, ya que no surgieron demasiados problemas en la programación y se consiguió que funcionara todo bastante rápido. De esta forma, se consiguió comenzar la segunda modificación prácticamente cuando estaba previsto.

La segunda modificación duró bastante más tiempo del previsto, ya que no se había pensado realizar la adaptación del algoritmo WLS-DC como se ha comentado antes. Además, esta adaptación llevó mucho tiempo ya que la implementación que se disponía del algoritmo Distance Contraction era bastante confusa y costó bastante tiempo conseguir corregir todos los errores y que el algoritmo funcionase correctamente.

La tercera modificación se implementó algo más rápido de lo esperado, ya que la programación de este bloque no ocasionó excesivos problemas. Por lo tanto la parte de programación se consiguió terminar relativamente en el tiempo que se había establecido inicialmente.

Las simulaciones llevaron más tiempo del esperado, ya que se han simulado más situaciones y algoritmos de los que se pensaba simular en un comienzo. La realización de la memoria se ha llevado a cabo en algo menos de dos meses, que era más o menos el tiempo previsto inicialmente.

4. Resultados

En este apartado se muestra un resumen de los resultados obtenidos en las simulaciones realizadas. En el anexo C, se muestran los resultados completos.

4.1 Comparativa de algoritmos básicos

4.1.1 Modelo estadístico

En primer lugar se van a evaluar los algoritmos de localización para el modelo estadístico del simulador. La evaluación de los algoritmos de localización se va a realizar en función del número de anchors utilizados para la estimación de la posición.

En la figura 10, se muestra el error medio de posicionamiento de los algoritmos, para una distancia entre anchors de 10 metros, lo que significa que el área está cubierta por 36 nodos anchor, y para un error residual de ranging de $\sigma_n = 0.7$.

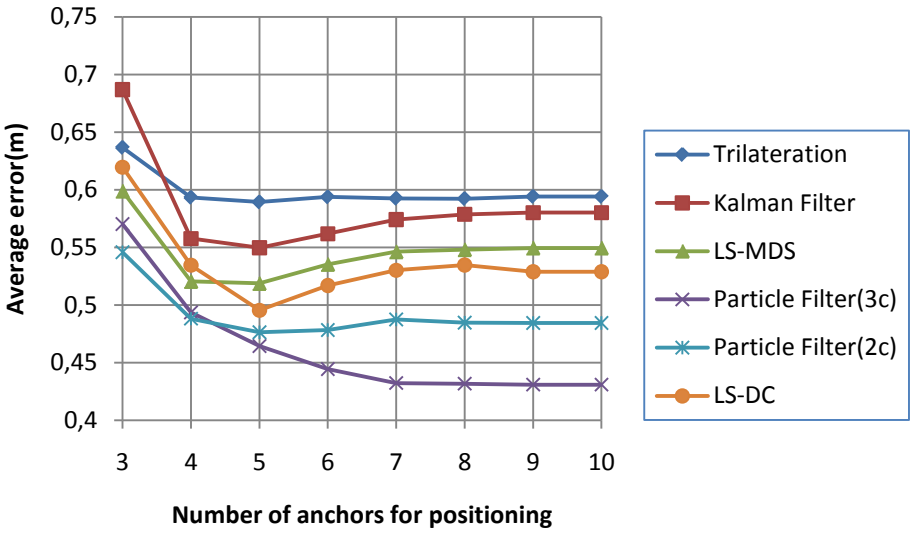


Figura 10. Error medio de posicionamiento. Distancia entre anchors = 10 m. σ_n = 0.7.

Como puede observarse en la figura 10, los mejores resultados se obtienen con el filtro de partículas. Para calcular la probabilidad de cada partícula se dispone de tres modelos, basados en una gaussiana, en la suma de dos gaussianas y en la suma de tres gaussianas respectivamente. Con la notación "3c" y "2c", se hace referencia al modelo usado, siendo "3c" el modelo basado en la suma de 3 gaussianas y "2c" el basado en la suma de dos gaussianas. No obstante, los resultados conseguidos con el filtro de partículas no son realistas, debido a que en el simulador las distancias estimadas se obtienen mediante un modelo de ranging estadístico. Y el filtro de partículas utiliza también un modelo estadístico del error de medida para obtener los pesos de las partículas. Además, ese modelo se ha optimizado mediante simulaciones, de tal forma que se comporta de manera similar al modelo de ranging. En un sistema real, una caracterización tan precisa del modelo de ranging especifico del escenario requeriría unas fases de medida y de calibración muy costosas. Y el uso de un modelo genérico no proporcionará resultados tan buenos. Por lo tanto, los resultados del filtro de partículas se deberan considerar como una referencia más que como una implementación realista. El

modelo de ranging estadístico tiene 3 componentes, por eso,el filtro de partículas utilizando el modelo de 3 componentes obtiene mejores resultados que utilizando solo dos componentes.

Los algoritmos LS-MDS, LS-DC y filtro de Kalman presentan una evolución similar, sin embargo los mejores resultados se obtienen con LS-MDS y LS-DC. En particular, con LS-MDS se obtienen mejores resultados para 3 y 4 anchors y con LS-DC para más de 4 anchors. En los tres algoritmos, el número óptimo de anchors es 5. Si se utilizan más anchors, los nuevos anchors estarán más lejos del target y tendrán mayor desviación de ranging, provocando que el error de posicionamiento aumente. Por último, la trilateración es el algoritmo que peores resultados ofrece y es independiente del número de anchors escogido, ya que únicamente se emplean los tres anchors más cercanos al target en el algoritmo.

Para todos los algoritmos hay un incremento del error cuando solo se utilizan tres anchors y el error se mantiene constante para más de 7 anchors, debido a que no es probable que el target esté en cobertura de más de 7 anchors.

Los resultados cuando se aumenta la distancia entre anchors a 12.5 metros, por lo que el área estará cubierta por 25 anchors, y se mantiene el error residual de ranging de $\sigma_n = 0.7$, se muestran en la figura 11.

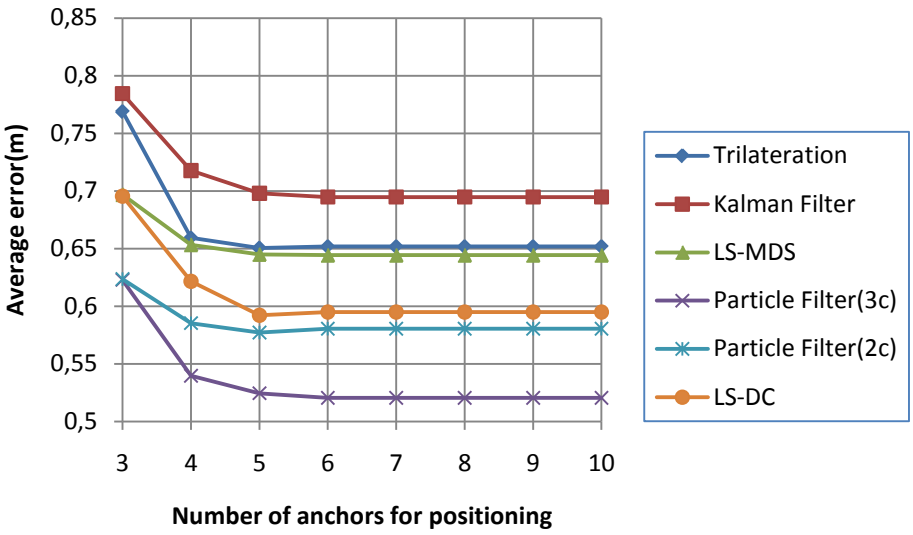


Figura 11. Error medio de posicionamiento. Distancia entre anchors = 12.5 m. σ_n = 0.7.

Debido a que la distancia entre anchors es elevada, los anchors utilizados para el posicionamiento estarán bastante lejos del target. El filtro de partículas presenta los mejores resultados ya que se beneficia de su preciso modelo del error de medida. El algoritmo LS-DC obtiene mejores resultados que el LS-MDS, ya que el algoritmo Distance Contraction funciona correctamente cuando la mayoría de las distancias estimadas a los anchors presentan una desviación positiva. Por lo tanto, cuanto mayor sea la distancia entre anchors, mayor será la desviación de las distancias estimadas y mejores serán los resultados frente al uso únicamente del MDS, es decir, frente al algoritmo LS-MDS. La trilateración y el LS-MDS tienen un comportamiento muy similar mientras que, los peores resultados se obtienen con el filtro de Kalman, ya que su simple modelo de error Gaussiano no puede tratar con las elevadas desviaciones de las distancias estimadas para anchors muy alejados.

Para todos los algoritmos el error se mantiene constante para más de 5 anchors, ya que no es probable que el target tenga cobertura con más de 5 anchors. No se consideran distancias mayores entre anchors, ya que se podría perder la cobertura entre los nodos anchor.

Ahora se muestran los resultados cuando el error residual de ranging disminuye a $\sigma_n = 0.3$, para una distancia entre anchors de 10 m, en la figura 12.

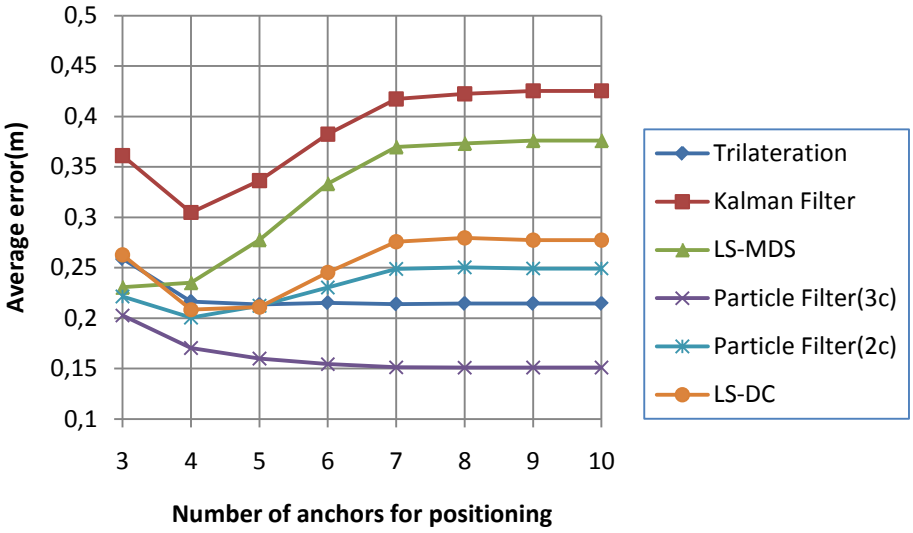


Figura 12. Error medio de posicionamiento. Distancia entre anchors = 10 m. σ_n = 0.3.

Como podría esperarse, el comportamiento de todos los algoritmos es mejor que en el caso de utilizar la misma configuración con un error residual de ranging de 0.7 y puede obtenerse un error medio de posicionamiento entorno a 15-30 cm. La mejora más remarcable es la del algoritmo de trilateración, que obtiene resultados comparables al LS-MDS, LS-DC y filtro de partículas de 2 componentes para un número de anchors igual a 4. Esto significa que la trilateración requiere unas estimaciones del TOA precisas para proporcionar buenos resultados, ya que siempre utiliza tres medidas para calcular la posición y no puede beneficiarse de la diversidad de las medidas. Como el error residual ha disminuido, las desviaciones del ranging tienen mayor importancia y por ello los anchors cercanos proporcionan estimaciones mucho más precisas que anchors alejados. Como consecuencia, el número óptimo de anchors es de 3 para el LS-MDS y de 4 para el filtro de Kalman, LS-DC y filtro de partículas de 2 componentes, empeorando mucho el comportamiento cuando se aumenta el número de anchors utilizados en el cálculo de la posición.

4.1.2 Modelo basado en paredes y rutas

En este punto se van a evaluar los algoritmos de localización para el modelo basado en paredes y rutas del simulador. Al igual que en el anterior apartado, la evaluación de los algoritmos de localización se va a realizar en función del número de anchors utilizados para el cálculo de la posición.

En la figura 13, se muestran los resultados para una distancia entre anchors de 10 m y un error residual de ranging de $\sigma_n = 0.7$. En la figura 5 se muestra el escenario modelado para esta configuración.

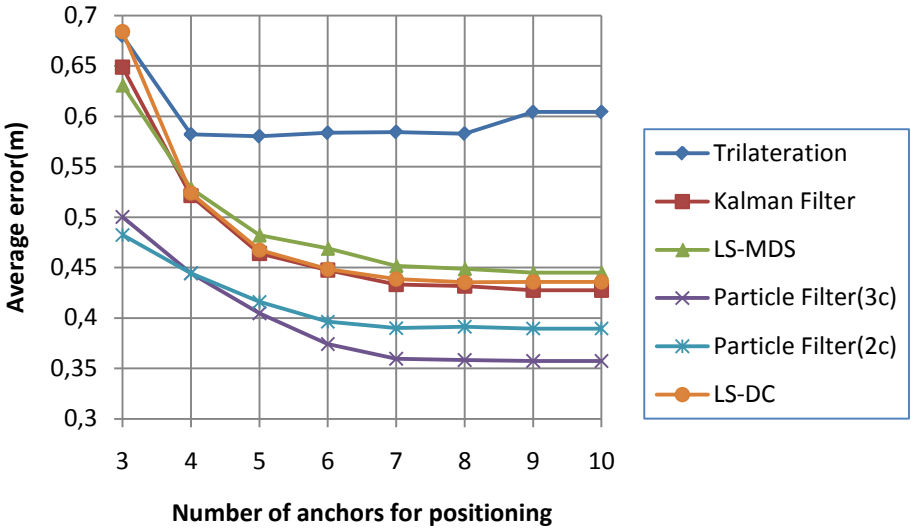


Figura 13. Error medio de posicionamiento. Distancia entre anchors = 10 m. σ_n = 0.7.

Los algoritmos LS-MDS, LS-DC y filtro de Kalman presentan un comportamiento muy similar, aunque el LS-MDS con un error mayor para 5 o más anchors. El filtro de partículas de 3 componentes obtiene mejores resultados que el de dos componentes para más de 4 anchors, ya que los anchors que se añaden están alejados y las desviaciones del ranging son mayores. La trilateración es el algoritmo con peores resultados, mientras que el filtro de partículas es el que ofrece los mejores resultados.

Para todos los algoritmos el error se mantiene constante para más de 7 anchors, debido a que no es probable que el target esté en cobertura de más de 7 anchors. Por lo tanto el número óptimo de anchors utilizados para el cálculo es 7.

Si se observa la figura 10, se puede ver que en el modelo estadístico para la misma configuración, el filtro de partículas obtiene mejores resultados respecto al resto de algoritmos que en el caso del modelo basado en paredes y rutas. Esto se debe a que se ha modificado el modelo de ranging y ya no se calcula la probabilidad de que el enlace entre target y anchor sea LOS, NLOS o NLOS severo sino que se calcula el número de paredes existentes entre el target y los anchors utilizados en la medida, y en base al número de paredes se determina la componente de error que se utiliza. El modelo estadístico del error de medida del filtro de partículas sigue ofreciendo muy buenos resultados, aunque en este caso se diferencia algo más del modelo de ranging.

En la figura 14 aparecen los resultados si se aumenta la distancia entre anchors a 12.5 m y se mantiene el error residual de ranging $\sigma_n = 0.7$. En la figura 6 se puede observar el plano modelado, para esta distancia entre anchors.

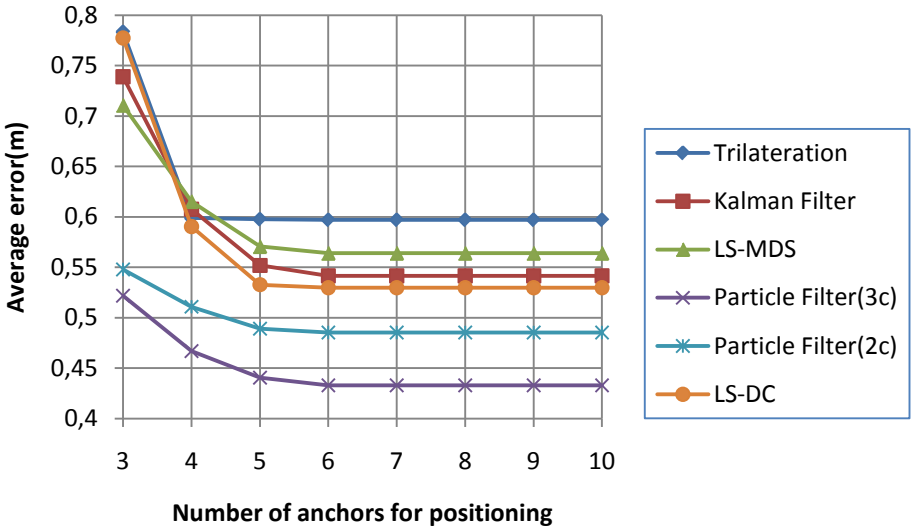


Figura 14. Error medio de posicionamiento. Distancia entre anchors = 12.5 m. σ_n = 0.7.

Ahora la distancia entre anchors es mayor, así que las desviaciones del ranging son mayores ya que los anchors están más alejados. Los algoritmos LS-MDS, LS-DC y filtro de Kalman presentan una evolución similar aunque el LS-MDS con un error mayor para más de 4 anchors. El LS-DC mejora más respecto al LS-MDS que en la configuración anterior, ya que como se explicó, el algoritmo DC funciona mejor cuando todas las distancias estimadas presentan un error positivo.

El filtro de partículas de 3 componentes es el que mejor resultados ofrece, ya que es capaz de compensar las grandes desviaciones del ranging para anchors alejados. El filtro de partículas de 2 componentes presenta una evolución similar aunque con un error bastante superior. La trilateración es el algoritmo que peores resultados ofrece.

Para todos los algoritmos el error se mantiene constante para más de 6 anchors, ya que no es probable que el target esté en cobertura de más de 6 anchors. Para esta configuración el número óptimo de anchors utilizados para el cálculo de la posición es 6.

Al igual que en el modelo estadístico no se consideran distancias mayores entre anchors, ya que se podría perder la cobertura entre los nodos anchor.

Ahora se presentan los resultados reduciendo el error residual de ranging a $\sigma_n = 0.3$. En la figura 15, se muestran los resultados para una distancia de 10 m entre anchors.

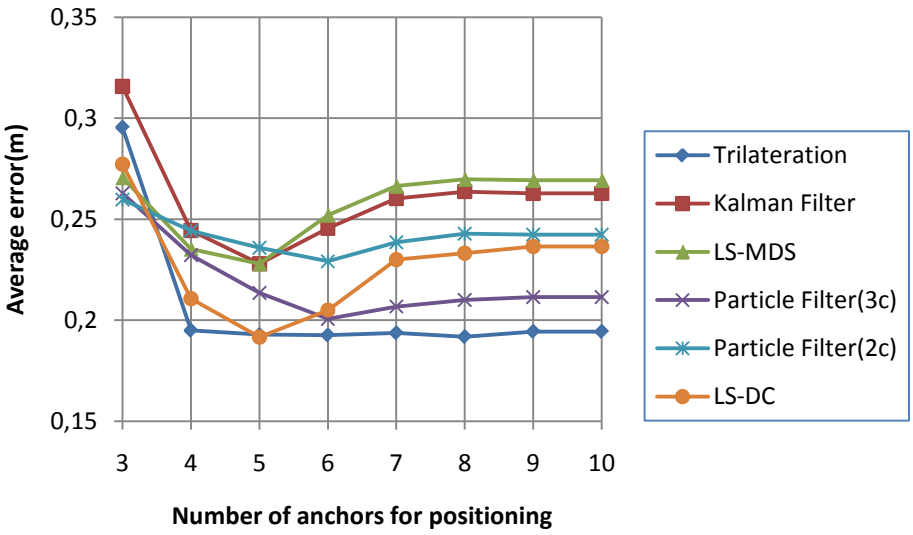


Figura 15. Error medio de posicionamiento. Distancia entre anchors = 10 m. σ_n = 0.3.

El comportamiento de todos los algoritmos es mejor que en la situación de utilizar la misma configuración para un error residual de ranging de 0.7. Como ya se vio en el modelo estadístico del simulador, como el error residual ha disminuido, las desviaciones del ranging tienen mayor importancia y por ello los anchors cercanos proporcionan estimaciones mucho más precisas que anchors alejados. Debido a esto, como la trilateración solo utiliza los tres anchors más cercanos, es el que mejor resultados ofrece para 4 o más anchors, ya que los anchors que se añaden proporcionan estimaciones mucho peores y hacen que el error de posicionamiento aumente, incluso para el filtro de partículas.

Los algoritmos LS-MDS, LS-DC y filtro de Kalman tienen una evolución similar, aunque el LS-DC ofrece mejores resultados. El número óptimo de anchors para los tres algoritmos es 5. Para los dos modelos del filtro de partículas el número óptimo de anchors es 6.

4.1.3 Filtro de partículas

Como se ha comentado previamente, los parámetros del filtro de partículas tanto para el modelo de 3 componentes como para el de 2 componentes se optimizaron mediante simulaciones. Sin embargo, en un sistema real los parámetros se fijarán mediante una fase de calibración que no ofrecerá resultados tan óptimos como mediante simulación. Por ello, en este apartado se van a comparar los resultados obtenidos por el filtro de partículas de 2 componentes optimizado, con el filtro de partículas de 2 componentes sin optimizar. La optimización se realizó para situaciones con un error residual de ranging de $\sigma_n = 0.7$ y una distancia entre anchors de 10 m. En la figura 16 se muestran los resultados de los filtros de 2 componentes con y sin optimización para el modelo basado en paredes y rutas del simulador en función del número de anchors utilizado para el cálculo de la posición.

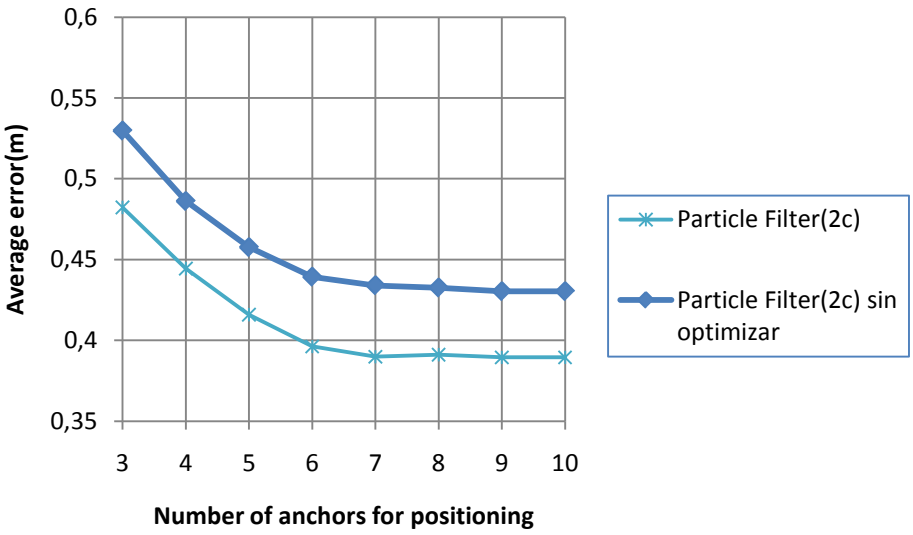


Figura 16. Error medio de posicionamiento. Distancia entre anchors = 10 m. σ_n = 0.7.

Como puede observarse, la evolución de ambos filtros de partículas es la misma, pero el optimizado obtiene mejores resultados. La diferencia entre los errores del filtro optimizado y del filtro sin optimizar se mantiene constante en torno a los 5 cm.

Por lo tanto, para disponer de un filtro de partículas perfectamente optimizado, se deberán optimizar los parámetros específicamente para el escenario en que se pretenda utilizar. Como ya se comentó, para conseguir una buena optimización, se requerirían unas fases de medida y de calibración muy costosas. Y si se utiliza un modelo genérico sin optimizar, se pierde precisión respecto a un modelo optimizado como se ha observado anteriormente en la figura 16.

4.2 Prefiltrado

A continuación se van a mostrar los resultados del prefiltrado de las estimaciones para la configuración de 10 m entre anchors y error residual de ranging $\sigma_n = 0.7$. Para evaluar el comportamiento de esta mejora se van a comparar diferentes estrategias de filtrado en función de la velocidad del target. Hay que tener en cuenta que la tasa de actualización de las distancias entre el target y los anchors es de 1 segundo. Los resultados se muestran en la figura 17.

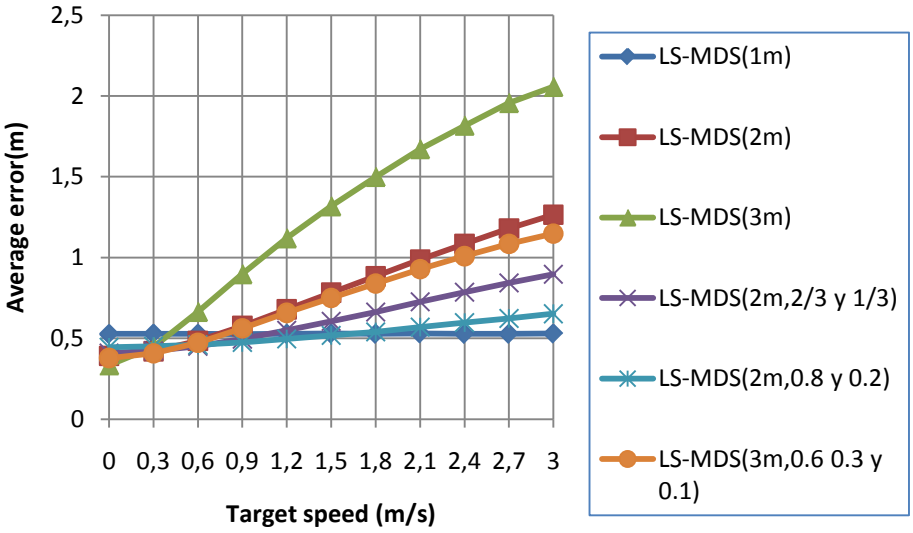


Figura 17. Error medio de posicionamiento. Distancia entre anchors =10 m. σ_n = 0.7.

En primer lugar se va a explicar la notación de la leyenda de la figura 17. El término 1m, 2m o 3m, se refiere al número de muestras que se utiliza para el filtrado. El caso 1m, significa que unicamente se utiliza la muestra actual, el caso 2m quiere decir que se utilizan la muestra actual y la muestra anterior, y el caso 3m significa que se utiliza la muestra actual y las dos anteriores. Si solo se nombra con 2m o 3m, quiere decir que se realiza la media entre las muestras. Mientras que si por ejemplo también se nombra con un 2/3 y 1/3, quiere decir que la muestra actual se pondera por 2/3 y la muestra anterior por 1/3.

Si el target se mueve a velocidades bajas, realizar un prefiltrado es beneficioso, llegando a reducir el error obtenido por el LS-MDS convencional, mientras que para velocidades altas, el prefiltrado empeora la situación, ya que hay mucha diferencia de distancia entre una muestra y la siguiente si el target se mueve muy rápido.

Una opción que funciona bastante bien es utilizar dos muestras y ponderar la actual por 0.8 y la anterior por 0.2, ya que obtiene mejores resultados para velocidades inferiores a 1.5 m/s. La opción de utilizar dos muestras pero ponderarlas con 2/3 y 1/3, obtiene mejores errores que la de 0.8 y 0.2, solo para velocidades inferiores a 0.4 m/s. Para velocidades superiores, el hecho de dar más valor a la muestra anterior hace que su comportamiento empeore.

Las opciones de realizar una media con dos muestras o utilizar tres muestras ponderandolas por 0.6, 0.3 y 0.1 obtienen resultados muy similares para velocidades inferiores a 1.5 m/s. Mientras que para velocidades superiores, la opción de dos muestras empeora ya que da menos peso a la muestra actual que la opción de tres muestras.

La peor opción es realizar la media con tres muestras, ya que la muestras actual y la muestra de dos iteraciones antes es muy diferente incluso para velocidades bajas. Y eso se ve en los resultados, ya que para velocidades superiores a 0.3 m/s el error medio se dispara.

Como conclusión podemos decir que el prefiltrado puede resultar beneficioso para velocidades inferiores a 1 m/s. Mientras que para velocidades superiores, una muestra y la siguiente se diferencian demasiado, provocando que el error medio de posicionamiento

empeore muy rapidamente. Por lo tanto se decidió no utilizar el prefiltrado en el simulador, ya que no compensa la mejora para velocidades bajas con el gran empeoramiento para velocidades algo más elevadas.

4.3 Ponderación

En este apartado se va a comprobar el funcionamiento de la adaptación de los algoritmos propuestos en PULSERS PHASE II (WLS-MDS) y EUWB (WLS-DC) incluyendo la ponderación de las medidas, en comparación con sus versiones sin ponderación (LS-MDS y LS-DC). Los algoritmos se van a comparar en función del número de anchors utilizados para estimar la posición.

En la figura 18 aparecen los resultados para una distancia de 10 m entre anchors y un error residual de ranging de $\sigma_n = 0.7$.

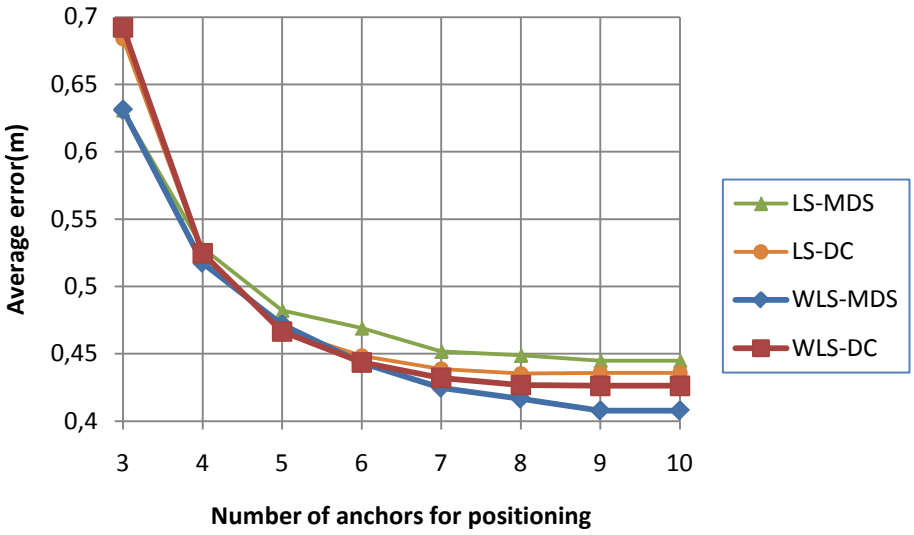


Figura 18. Error medio de posicionamiento. Distancia entre anchors = 10 m. σ_n = 0.7.

Primero vamos a analizar el algoritmo LS-MDS. La adaptación del algoritmo PULSERS PHASE II (WLS-MDS) funciona bastante bien para más de 4 anchors, consiguiendo reducir el error medio de posicionamiento mediante la ponderación de las distancias estimadas en base a su variabilidad. Los anchors que se añaden a partir de 4 anchors, son anchors alejados y por lo tanto sus estimaciones tienen una mayor variabilidad que anchors cercanos, por ello la ponderación es capaz de compensar esa variabilidad y mejorar la precisión.

Ahora analizamos el algoritmo LS-DC. La adaptación del algoritmo propuesto en EUWB (WLS-DC), obtiene los mismos resultados que el LS-DC salvo para más de 6 anchors que obtiene unos errores medios algo inferiores. En este algoritmo la ponderación de las distancias no funciona tan bien como para el caso del LS-MDS.

Aunque para esta configuración el algoritmo DC mejoraba respecto al uso únicamente del MDS (LS-MDS), los pesos se comportan mejor para el LS-MDS haciendo que el algoritmo que mejores resultados obtiene sea el WLS-MDS.

Ahora vamos a observar los resultados pero reduciendo el error residual de ranging a $\sigma_n = 0.3$. En la figura 19 aparecen los resultados para una distancia entre anchors de 10 m.

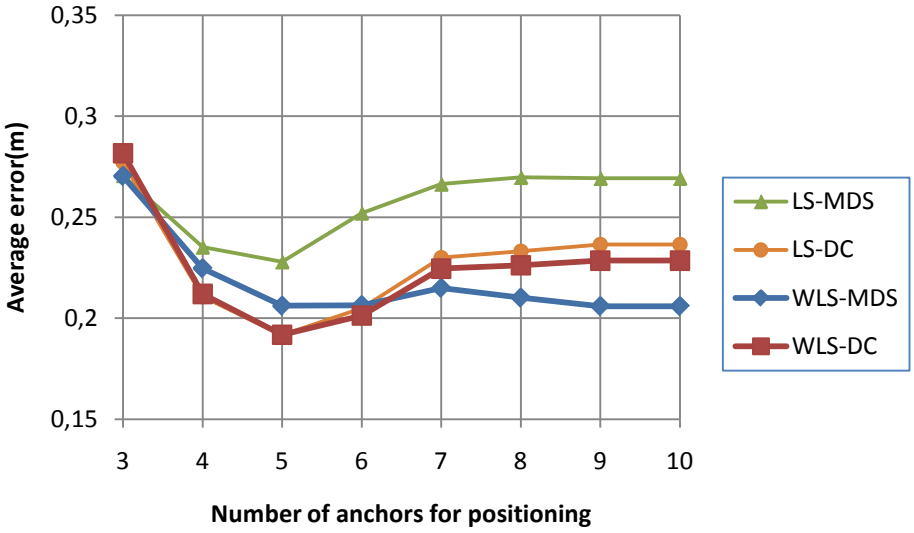


Figura 19. Error medio de posicionamiento. Distancia entre anchors = 10 m. σ_n = 0.3.

Al reducirse el error residual, las desviaciones del ranging son más importantes y los anchors cercanos ofrecen estimaciones mucho más precisas que los anchors alejados. Todos los algoritmos se comportan mejor que en la configuración con error residual de 0.7. El número óptimo de anchors para todos los algoritmos es 5. El WLS-MDS consigue reducir bastante el error respecto al LS-MDS.

El WLS-DC obtiene los mismos errores que el LS-DC para 5 o menos anchors, mientras que para más de 5 obtiene errores levemente inferiores.

4.4 Uso de información geográfica

4.4.1 WLS-MDS y WLS-DC

En este apartado se va a comprobar el funcionamiento de la modificación de identificación de situaciones NLOS, para los algoritmos WLS-MDS y WLS-DC (WLS-MDS & NLOS y WLS-DC & NLOS). En estos algoritmos la identificación de situaciones NLOS consistía en comprobar si el enlace entre el target y los anchors utilizados estaba en situación de LOS o de NLOS comprobando si el enlace cruzaba alguna pared del plano. Si el enlace estaba en situación de NLOS, el peso correspondiente a ese enlace se multiplicaba por 0.5.

En la figura 20 aparecen los resultados para una distancia de 10 m entre anchors y un error residual de ranging de $\sigma_n = 0.7$.

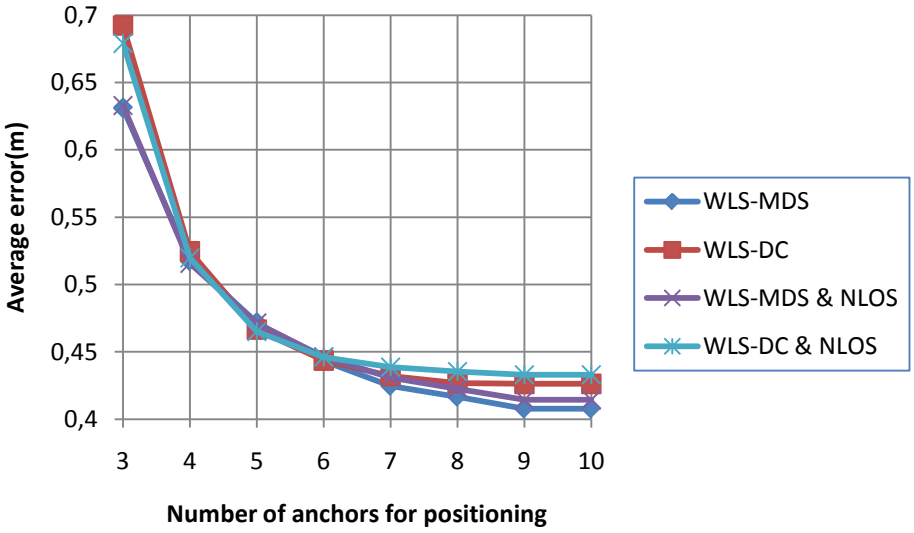


Figura 20. Error medio de posicionamiento. Distancia entre anchors = 10 m. σ_n = 0.7.

El algoritmo WLS-MDS & NLOS sigue la misma evolución que el WLS-MDS, obteniendo los mismos errores para 6 o menos anchors, aunque obtiene errores mayores para más de 6 anchors. La modificación de la identificación de situaciones NLOS no funciona como se esperaba, ya que se esperaba que funcionase un poco mejor que el WLS-MDS y es debido a que con la ponderación ya se está dando menos peso a situaciones de NLOS, ya que estas presentan mayor variabilidad que las situaciones de LOS, y al multiplicar el peso por 0.5, se está reduciendo el peso de las situaciones NLOS en exceso, provocando ese aumento del error respecto al WLS-MDS para un número de anchors superior a 6.

En cuanto al algoritmo WLS-DC & NLOS, tiene la misma evolución que el WLS-DC, obteniendo los mismos errores para 6 o menos anchors y errores algo superiores para más de 6 anchors.

Ahora vamos a observar los resultados reduciendo el error residual de ranging a $\sigma_n = 0.3$. En la figura 21 aparecen los resultados para una distancia entre anchors de 10 m.

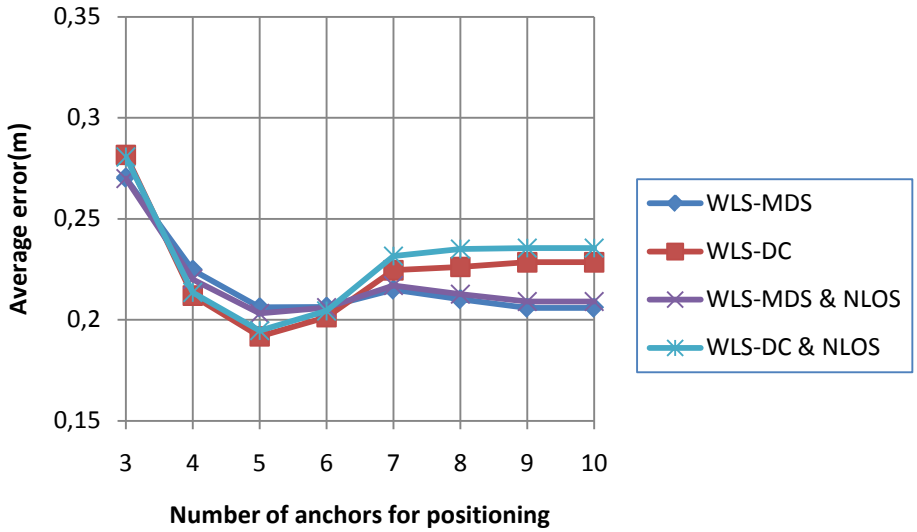


Figura 21. Error medio de posicionamiento. Distancia entre anchors = 10 m. σ_n = 0.3.

Al reducirse el error residual, las desviaciones del ranging son más importantes y los anchors cercanos ofrecen estimaciones mucho más precisas que los anchors alejados. Los algoritmos se comportan mejor que en la configuración con error residual de 0.7. El número óptimo de anchors para todos los algoritmos es 5. El WLS-MDS & NLOS obtiene errores practicamente idénticos a los obtenidos por el WLS-MDS.

El WLS-DC & NLOS obtiene los mismos errores que el WLS-DC para 6 o menos anchors, mientras que para más de 6 obtiene errores algo superiores.

4.4.2 Filtro de partículas

Ahora vamos a comparar los resultados obtenidos con el filtro de partículas de 3 componentes del modelo basado en planos y rutas del simulador, con los obtenidos con detección de situaciones NLOS y utilización de información acerca de las rutas de los target. La detección de situaciones NLOS en el filtro de partículas consistía en modificar el modelo de cálculo de la probabilidad de las partículas. Sin detección, los pesos LOS, NLOS y NLOS2 (severe NLOS), se calculan de forma estadística, mientras que con detección de situaciones NLOS, si no existen paredes en el enlace, el peso LOS toma valor uno, si hay una pared toma valor uno el peso NLOS y si hay dos o más paredes toma valor uno el peso NLOS2.

La utilización de información acerca de las rutas consistía en dos modificaciones. La primera consistía en dividir la probabilidad de la partícula entre 6, si estaba fuera de las rutas posibles. La segunda modificación consistía en que cuando el target se encontrase en un nodo, se modificaban las direcciones del 80% de las partículas de acuerdo a las probabilidades determinadas en el fichero likelihood para seguir recto, girar a la derecha, girar a la izquierda o dar media vuelta.

En la figura 22 se observan los resultados para una distancia entre anchors de 10 m y un error residual de ranging de 0.7. El algoritmo Particle Filter & NLOS & routes es el filtro de partículas con detección de situaciones NLOS y utilización de información acerca de las rutas.

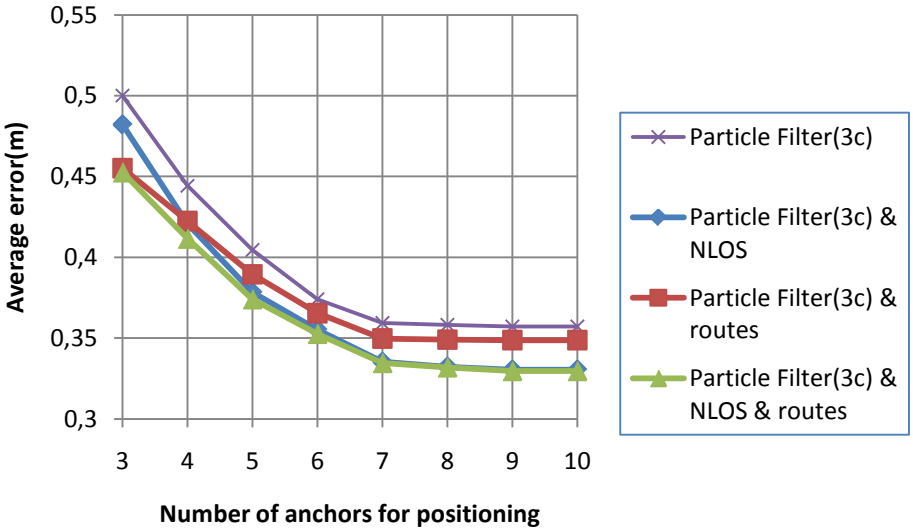


Figura 22. Error medio de posicionamiento. Distancia entre anchors = 10 m. σ_n = 0.7.

Los cuatro algoritmos siguen una evolución similar y es el filtro de partículas con ambas modificaciones el que mejor resultados ofrece. Viendo los resultados se puede concluir que la modificación que más mejora la precisión es la detección de situaciones NLOS, ya que obtiene errores inferiores para más de 3 anchors, que la utilización de información de las rutas. Además, para más de 5 anchors el filtro con detección de situaciones NLOS obtiene los mismos resultados que el filtro que combina ambas modificaciones. Es lógico que la detección de situaciones NLOS funcione mejor para un número de anchors elevado, ya que cuantos más anchors se utilicen, mayor es la probabilidad de que haya varios enlaces en situación de NLOS.

Ahora se van a analizar los resultados reduciendo el error residual de ranging a 0.3. En la figura 23 se muestran los resultados para una distancia de 10 m entre anchors.

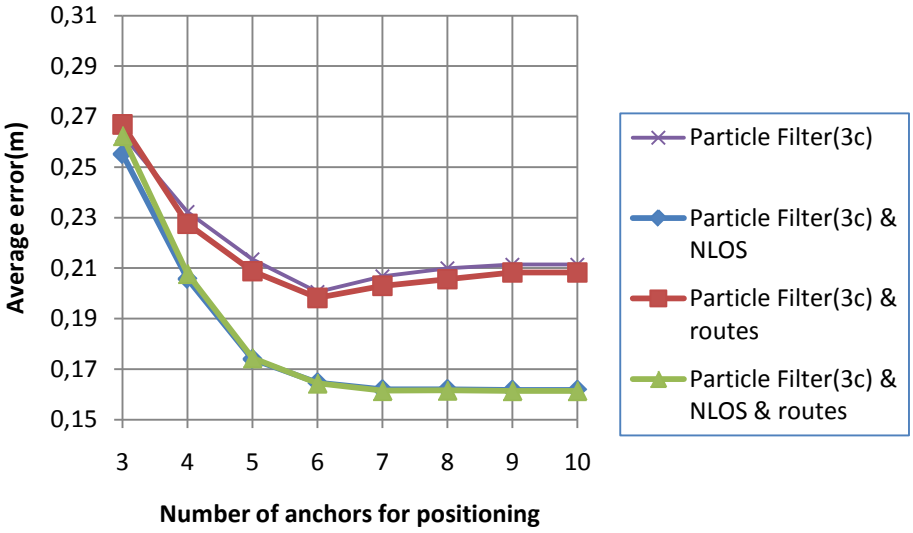


Figura 23. Error medio de posicionamiento. Distancia entre anchors = 10 m. σ_n = 0.3.

Como el error residual es inferior, los resultados obtenidos son mejores que en la situación anterior. Como ahora las estimaciones de los anchors cercanos son mucho más precisas que las de los anchors alejados, el número óptimo de anchors para el filtro convencional y para el filtro con la modificación de las rutas es de 6, mientras que para los otros dos filtros es de 7. La modificación de las rutas apenas afecta, ya que los errores obtenidos con la modificación son prácticamente idénticos a los obtenidos con el filtro convencional.

Los mejores resultados se obtienen con el filtro con la modificación de NLOS y con el filtro que tiene implementadas ambas. Ambos filtros obtienen los mismos errores, ya que la modificación de rutas no afecta prácticamente. Cuanto mayor es el número de anchors, mejor funciona la modificación de NLOS, ya que es más probable que haya varios enlaces en condiciones de NLOS. Otra ventaja de la detección de situaciones NLOS es que ya no son necesarios los parámetros relacionados a la probabilidad de LOS/NLOS/NLOS 2, con lo que la fase de optimización es menos costosa y el comportamiento real del filtro será más parecido al comportamiento en simulación.

4.4.3 Filtro de Kalman

En este punto se van a observar los resultados de las modificaciones de detección de situaciones NLOS y de utilización de información acerca de las rutas de los targets para el filtro de Kalman. La detección de situaciones NLOS consiste en multiplicar por 1.5 el valor de covarianza cuando se construye la matriz de covarianza del ruido de medida R. En cuanto a la utilización de información acerca de las rutas, consiste en que cuando el target está en un nodo, se multiplica el valor de la aceleración por 5 cuando se construye la matriz Q. Se van a analizar varias configuraciones del modelo basado en planos y rutas del simulador, en función del número de anchors utilizados para el cálculo de la posición. En la figura 24 se muestran los resultados obtenidos para una distancia de 10 m entre anchors y un error residual de ranging de $\sigma_n = 0.7$.

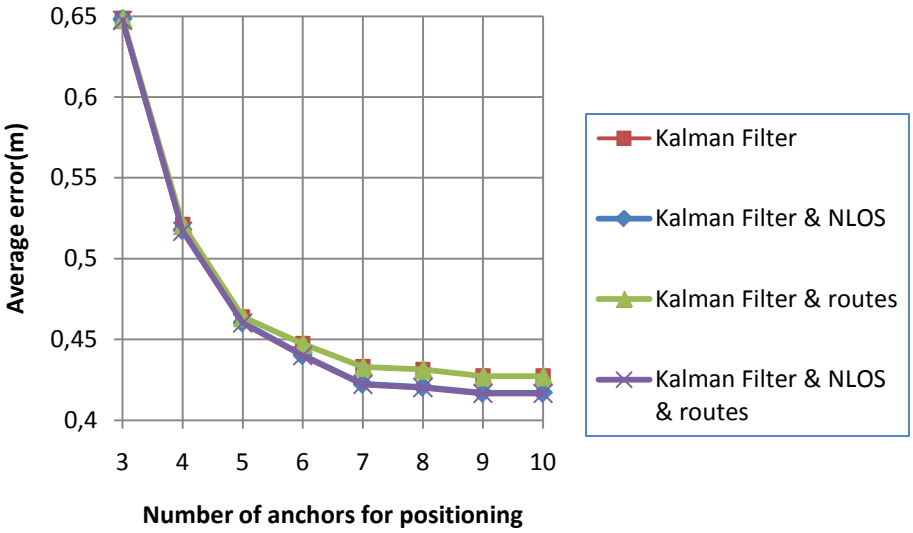


Figura 24. Error medio de posicionamiento. Distancia entre anchors =10 m. σ_n = 0.7.

Los cuatro algoritmos siguen una evolución similar, aunque para más de 5 anchors, el filtro con la modificación de detección de NLOS y el filtro con ambas modificaciones consiguen errores más pequeños. La modificación de las rutas no modifica apenas el funcionamiento del filtro de Kalman, ya que obtiene prácticamente los mismos errores que el filtro de Kalman convencional. La modificación de la detección de situaciones NLOS funciona mejor al aumentar el número de anchors, ya que aumenta el número de anchors que se encuentran en situación de NLOS. Como la modificación de las rutas apenas modifica el funcionamiento, el filtro con la modificación de NLOS obtiene los mismos resultados que el filtro con ambas modificaciones.

A continuación se presentan los resultados obtenidos disminuyendo el error residual de ranging. En la figura 25 se muestran los resultados para una distancia entre anchors de 10 m y un error residual de 0.3.

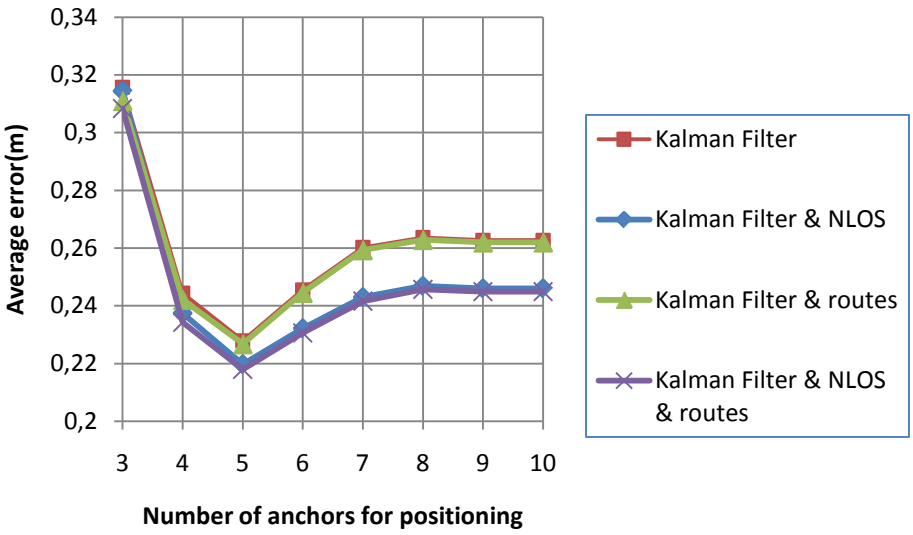


Figura 25. Error medio de posicionamiento. Distancia entre anchors =10 m. σ_n = 0.3.

Como se ha reducido el nivel de error residual, los anchors cercanos ofrecen estimaciones mucho más precisas que los anchors lejanos, por ello el número óptimo de anchors es 5 para todos los algoritmos, ya que añadiendo más de 5 anchors, se añaden anchors alejados y sus estimaciones tienen una gran desviación de ranging, provocando que el error medio de posicionamiento aumente. Al igual que en la misma configuración para error residual de 0.7 (figura 24), la modificación de las rutas no mejora el funcionamiento del filtro de Kalman convencional, obteniendo los mismos errores. La modificación de la detección de situaciones NLOS mejora el error del filtro de Kalman convencional para más de 3 anchors.

A modo de resumen, se puede concluir que la mejora de utilización de información acerca de las rutas apenas modifica el funcionamiento del filtro de Kalman y no consigue mejorar el error. Sin embargo la modificación de detección de situaciones NLOS consigue mejorar el funcionamiento del filtro de Kalman. Esta modificación funciona mejor para un número elevado de anchors. El filtro con ambas modificaciones implementadas obtiene los mismos resultados que el filtro con detección de situaciones NLOS, lo cual es lógico, ya que la modificación de las rutas no produce cambios en el comportamiento del filtro convencional.

5. Conclusiones

5.1 Conclusiones del proyecto

El objetivo de este proyecto era evaluar distintos algoritmos de localización para su uso en un sistema de seguimiento en interiores basado en tecnología UWB, además de proponer diversas mejoras basadas en el uso de información geográfica y estadística. Con el fin de observar el funcionamiento de los algoritmos, se han realizado simulaciones para diferentes situaciones y se han valorado y comentado en el apartado de resultados.

Para poder implementar y evaluar las mejoras de los algoritmos fue necesario adaptar el simulador ya existente, basado en información estadística, a un escenario basado en paredes y rutas preestablecidas. En base a ambos escenarios se han evaluado los algoritmos considerados. La trilateración y el filtro de Kalman son algoritmos sencillos aunque presentan bastantes limitaciones, en particular la trilateración es muy sensible al error de ranging residual, por lo que requiere una elevada resolución en la estimación del TOA, mientras que el filtro de Kalman es muy sensible a las desviaciones en la estimación que se producen en anchors alejados o en situaciones de NLOS, por lo que presenta malos resultados para distancias entre anchors elevadas o niveles de error residual bajo. El filtro de partículas presenta por lo general los mejores resultados, aunque esto es debido a la gran similitud del modelo de error en las observaciones utilizado en el filtro y el modelo utilizado para generar el error de ranging, especialmente en el caso del modelo de 3 componentes optimizado mediante simulaciones. Sin embargo, en una situación real el modelo debería caracterizarse a través de una fase de calibración cuyos resultados no serán óptimos, con lo que su precisión real será menor, como puede observarse cuando se utiliza el modelo de 2 componentes sin optimizar. Los algoritmos propuestos en el marco de los proyectos PULSERS y EUWB, basados en MDS y DC respectivamente, presentan buenos resultados en todos los escenarios, con la ventaja añadida en el caso de DC de que es capaz de corregir en cierta medida los errores debido a desviaciones en caso de NLOS, por lo que tiene un mejor comportamiento en casos con distancia entre anchors elevada o error residual bajo.

Un primer bloque de mejoras implementadas, propuestas en el proyecto EUWB para MDS y DC, fueron el prefiltrado y la ponderación de las estimaciones. El prefiltrado permite mejorar la precisión en el caso de que la velocidad de los móviles sea inferior a 1 m/s para el caso de tomar una medida por segundo, aunque se degrada de manera importante para velocidades superiores, por lo que no resulta muy adecuado para el seguimiento de targets móviles. Por lo que respecta a la ponderación, consiste en la aplicación de una serie de pesos en el algoritmo de optimización utilizado (SMACOF) calculados en base a la varianza de las estimaciones, con lo que se consigue ponderar en mayor medida las estimaciones más precisas, correspondientes a los anchors más cercanos y en visión directa. El uso de ponderaciones mejora en gran medida las prestaciones de MDS, con lo que el algoritmo WLS-MDS presenta un mejor comportamiento que el DC. Sin embargo, en el caso de DC, el algoritmo WLS-DC no presenta ninguna mejora frente a LS-DC, ya que el propio algoritmo DC se encargaba de compensar las desviaciones de los anchors alejados o en NLOS.

La principal aportación del proyecto se centra en la utilización de información geográfica, con el objetivo de mejorar la precisión de los algoritmos. La primera mejora consiste en identificar cuándo un enlace entre el target y uno de los anchors está en condiciones de NLOS y en aprovechar esa información. Esta información se utiliza para modificar el modelo de error de las observaciones en el caso del filtro de partículas y del filtro de Kalman, y la ponderación en el caso de WLS-MDS y WLS-DC. Esta modificación obtuvo bueno resultados especialmente en el filtro de partículas, ya que permite identificar la componente de error a utilizar (LOS/NLOS/NLOS2) en lugar de utilizar una suma ponderada de todas, además de que simplificaría la fase de calibración. Por otro lado, en los algoritmos WLS-MDS y WLS-DC no supuso ninguna mejora, ya que la ponderación ya identificaba las situaciones de NLOS. La segunda mejora consiste en utilizar información acerca de las rutas de los targets. De cara a aprovechar esta información, se modificaron los modelos dinámicos en el filtro de Kalman y en el filtro de partículas. En el filtro de partículas se consiguió mejorar la precisión en algunas situaciones, mientras que en el filtro de Kalman no supuso ninguna mejora.

5.2 Mejoras del proyecto

En este punto se van a exponer diversos proyectos que se podrían llevar a cabo en un futuro como continuación de este proyecto fin de carrera.

Una posible vía de trabajo sería probar los algoritmos propuestos en este proyecto en dispositivos físicos reales, para poder evaluar el comportamiento de los mismos en una situación real. Habría que montar una red de anchors en un determinado escenario y posteriormente modelar ese escenario para poder aprovechar la información geográfica, como se ha hecho con los modelos utilizados en este PFC. También se debe disponer de nodos que actúen como targets, para moverlos por el escenario y realizar medidas para comprobar el comportamiento de los diferentes algoritmos.

Otra vía de trabajo sería evaluar la carga computacional de los algoritmos. Un aspecto muy importante a la hora de evaluar algoritmos es la precisión obtenida, pero tampoco nos podemos olvidar de la carga computacional que tiene cada uno, ya que estos algoritmos se van a implementar en dispositivos portátiles, que tendrán una capacidad computacional limitada. Se trataría de evaluar el tiempo de ejecución de los diferentes algoritmos y comprobar que los dispositivos son capaces de computar la posición del target, antes de que se estimen de nuevo las distancias con los anchors. Para este estudio sería necesario conocer en profundidad la arquitectura de los dispositivos que se van a emplear.

5.3 Consideraciones personales

La realización de este PFC me ha aportado mucha experiencia, tanto desde el punto de vista de aprender un nuevo lenguaje de programación como es C++, como desde el punto de vista de trabajar en el I3A, que ha sido una experiencia muy enriquecedora para mi futuro laboral. Por último, para poner el punto final a esta memoria, quiero agradecer a Juan, director del proyecto, por su disponibilidad, por su experiencia y por su forma de dirigir este proyecto, que me ha parecido perfecta. También quiero agradecer a Toni y a Ángela, por su confianza depositada en mí y por su ayuda prestada.

6. Referencias bibliográficas

[1] J. Chóliz, A. Hernández, A. Valdovinos, "Architectures for Location Data Acquisition and Distribution in UWB Indoor Tracking Systems", In Proc. 7th Workshop on Positioning, Navigation and Communications (WPNC'10), Marzo 2010.

[2] J. Chóliz, A. Hernández, A. Valdovinos, "Evaluation of Algorithms for UWB indoor tracking", sin publicar.

[3] M. C. Almolda "Evaluación de algoritmos y estrategias de localización en sistemas UWB", Proyecto Fin de Carrera, Universidad de Zaragoza, Febrero 2010.

[4] G. Welch, G.Bishop, "An Introduction to the Kalman Filter", University of North Carolina at Chapel Hill, Julio 2006.

[5] F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, P. Nordlund, "Particle Filters for Positioning, Navigation, and Tracking", IEEE Transactions on signal processing, Vol. 50, Nº 2, Febrero 2002.

[6] P. Nordlund, F. Gunnarsson, F. Gustafsson, "Particle Filters for Positioning in Wireless Networks", In Proceedings of the XI European Signal Processing Conference (EURSIPCO'02), Septiembre 2002.

[7] G. Destino, D. Macagnano, G. Abreu, R. Zeltik, W. Kotterman, R. Thomae, S. Severi, D. Dardari, V. Latosa, "Initial Localization and Tracking Algorithm", Integrated Project EUWB, Deliverable D4.1.1, Junio 2009.

[8] G. Destino, D. Macagnano, G. Abreu, J. Chóliz, A. Hernández, R. Zeltik, G. Shen, V. Latosa, "Enhanced LT algorithms with heterogeneous information – initial", Integrated Project EUWB, Deliverable D4.1.2a, Mayo 2010.

[9] J. de Leeuw, P.Mair, "Multidimensional Scaling Using Majorization: SMACOF in R", Septiembre 2008.

[10] A. M. Johansen, "SMCTC: Sequential Monte Carlo in C++", Journal of Statistical Software, 30(6):1-41, Abril 2009.

[11] A. Álvarez, L. de Celis, G. Destino, D. Xu, S. Wang, R. Zeltik, "Implementation of the enhanced LT engine with mobility management (LDR and HDR)", Integrated Project EUWB, Deliverable D4.3.2a, Octubre 2009.

[12] H. Schildt, "The complete reference C++", McGraw-Hill, 1998.